UNIVERSITY OF CALIFORNIA

SANTA CRUZ

ACCURATE GENOME ANALYSIS WITH NANOPORE SEQUENCING USING DEEP NEURAL NETWORKS.

A dissertation submitted in partial satisfaction of the

requirements for the degree of

DOCTOR OF PHILOSOPHY

in

BIOMOLECULAR ENGINEERING AND BIOINFORMATICS

by

Kishwar Shafin

March 2022

The Dissertation of Kishwar Shafin
is approved:

Professor Benedict Paten, Chair

Professor David Haussler

Professor Mark Akeson

Peter F. Biehl
Vice Provost and Dean of Graduate Studies

Contents

List of Figures

List of Tables

Abstract

Accurate genome analysis with nanopore sequencing using deep neural networks.

by

Kishwar Shafin

Nanopore sequencing, commercialized by Oxford Nanopore Technology (ONT), is a high-throughput genome sequencing platform. Unlike traditional sequencing-by-synthesis methods, nanopore sequencing uses measured current signals to sense the nucleotide sequence flowing through the pore. The signal-to-base conversion process introduces unique error patterns, making it challenging to design methods that rely on hand-crafted features. Deep learning uses multiple layers to progressively learn complex patterns in the input data, making it suitable for genome analysis. In this dissertation research, I present methods I developed based on deep neural networks to improve genome inference with nanopore sequencing. First, I introduce haplotype-aware variant calling pipeline PEPPER-Margin-DeepVariant that produces state-of-the-art results for nanopore long-reads. Next, I demonstrate a pipeline to perform *de novo* assembly of eleven human genomes in nine days. Then I show the application of the methods to validate and correct errors in the first complete human genome assembly. Finally, I demonstrate the utility of PEPPER-Margin-DeepVariant paired with highly multiplexed nanopore sequencing for rapidly identifying disease-causing variants.

To my wife,

Sanaiya Islam,

Among infinite universes, there doesn't exist one where I achieved this without your support.

Acknowledgments

I want to thank my committee members, Benedict Paten, Mark Akeson, and David Haussler, for taking the time to assess the work I presented in my dissertation. I also want to thank Kevin Karplus, Olena Vaske, Manual Ares for being on my advancement committee. A special thanks to Benedict Paten for the years of mentoring he has provided. Benedict's mentorship and trust in me to solve some most challenging problems has made my journey rewarding.

Five years ago, I moved 7,731 miles from Dhaka, Bangladesh, to Santa Cruz to pursue the highest level of education in bioinformatics with zero previous experience. The move that was near twice the size of the earth's radius didn't scare me that much, but getting up to speed within the first year of my graduate program with no background felt like being haunted in a horror movie. I was terrified, had no control over anything, and all I wanted was everything to stop. It often felt that I could not do this, and graduate school was not for me. The alternatives were even scarier as an international student. However, after five years and four months, when I am defending my dissertation, I am enormously proud and happy that I did this. I would not trade my graduate school experience with anything. I want to thank David Bernick and Richard E. Green for providing me with the cushion I needed at that time and my family and friends for saying all the good things you always say. For once, those made a pretty big difference.

As an international student in Santa Cruz, I never felt like I was too far away from home.

Bryan Thornlow, Brandon Saint-John, Roger Volden, Colleen Bosworth, Jon Akutagawa, Akshar Lohith, and Alison Tang thank you for making Santa Cruz feel like home.

I had a great experience at Computational Genomics Lab. I am grateful to have worked with Marina Haukness, Ryan Lorig-Roach, Jordan Eizenga, Melissa Meredith, Hugh Olsen, Xian Chang, Adam Novak, Robin Rounthwaite, Cecilia Cisar, and Mobin Asri. Working with such a vibrant group of people was a privilege. I would also like to thank Louise Cabansay for helping me get started with my research.

One person I owe a lot of gratitude to is Trevor Pesout. Graduate school felt like being a part of a team because of Trevor. I could not ask for a better colleague and friend. Thank you for stepping up every time, work-related or not.

Thank you, Miten Jain, for your support as a friend and colleague; you have greatly supported this journey. I am also grateful to Paolo Carnevali from CZI for guiding me in many problems related to genome assembly.

I think I was the luckiest in terms of having the best collaborators. Andrew Carroll and Pi-Chuan Chang, I can’t thank you enough for countless support and help. It has been the most fulfilling experience. I want to thank Maria Nattestad, Daniel Cook, Gunjan Baid, and Ted Yun for welcoming me into their team during my internship at Google.

The telomere-to-telomere consortium was an exceptional experience of my graduate life. I am humbled to be a part of the T2T consortium. I want to thank Karen Miga, Adam

Phillippy, Arang Rhie, Michael Alonge, Ann McCartney, Giulio Formenti, Arkarachai Fungtammasan, Samantha Zarate, Sergey Koren, Justin Zook, Michael Schatz for helping me to learn quickly.

I want to thank Euan Ashley, Sneha Goenka, John Gorzynski, and the entire Stanford Clinical team for allowing us to take our pipelines closer to the patients. The amount of trust this group has shown on our work is tremendous. I really did not expect to have real life impact of our research during graduate school. I am also grateful to the patients who trusted our research. I want to thank Johnny Israeli and Eric Dawson for the internship opportunity at NVIDIA.

Last but not least, thanks to my family. My mom Begum Saleha and dad Abul Kalam, thank you for supporting me throughout my journey. My sister Sazia Sabrin and niece Mahdia Maliyat, thank you, it means a lot to go back to a supporting family. Sanaiya Islam, thank you for constantly reminding me to be the best version of myself and reminding me that life is an experience that can not be measured in successes and failures.

Part I

Introduction

Chapter 1

Introduction

Whole-genome sequencing has enabled the comprehensive study of the genome to answer a more extensive set of questions. Majority of the experiments in human genomics are centered around understanding the genomic variation between samples or populations of samples. Studying genomic variation at the whole genome scale is more potent than microarray-based methods that only look at a handful of variations. Microarray-based methods have limited application and often suffer to provide a scientific understanding through a handful of variations. On the other hand, whole-genome sequencing aims to unravel every nucleotide base, providing a way to study the genome comprehensively. Since the early 2000s, as genome sequencing got cheaper and more accessible, we saw whole-genome analysis helping to understand biological processes, diagnose rare diseases, and uncover the evolutionary history through the lens of population genetics.

Our ability to sequence and analyze genomic variation largely depends on a set of resources we build around it, known as reference genome materials. The reference genome is a genome assembly of a sample of the same species that we consider representative of the species. The genome reference consortium (GRC) has created a human reference genome available that we can use to study human genomes. As a species, we humans have sequenced an extensive amount of human genomes; however, the quality of the inference and our understanding of the human genome still depends on what technology we use for genome sequencing and which reference we use.

Next-generation short-read sequencing technology provided an affordable solution to genome inference. However, mapping the short reads to the repetitive regions of the genome remains challenging. It requires assembling a large cohort of human genomes to create a haplotype map so short reads can use the human genome variation to map to the right haplotype, which is an expensive endeavor. In short, short-read sequencing is unable to provide *de novo* genome assemblies that can span through the repeat-rich regions of the genome, limiting researchers to study the difficult regions of the genome, analyze large structural variations or map to the repetitive regions of a linear reference to provide a complete picture of the genomic variations.

Recently, nanopore long-read sequencing technology promised the ability to generate contiguous *de novo* assemblies, map through the repetitive regions of the genome, provide a comprehensive view of the structural variation that significantly improves our ability to study the genome. Nanopore sequencing uses sensed electrical signals to derive the nucleotide base sequence going through the pore. This process is fundamentally different than sequencing-by-synthesis and has unique error modalities in homopolymer regions. Although nanopore long-read sequencing can potentially change how we study human genomes, designing genome inference methods requires deriving a complex set of features that correctly capture the error modalities.

Deep learning provides a set of algorithms that can automatically learn complex features of the input data. The ability to seamlessly learn intricate abstractions has made deep learning algorithms a success in fields that deal with large and complex data, and biology generates one of the most complex data sets. The difficulty in designing deep learning methods for long-read sequencing lies in understanding the error modalities and representing them correctly so the model can learn from the data.

My research has focused on developing methods to improve genome inference with nanopore sequencing technology. Initially the research seemed highly ambitious as the initial error of nanopore was around 7%, and many were skeptical about its ability to be go beyond microbial genomes and used on complex human genomes in a clinical setting. Having an accurate variant calling and *de novo* assembly method meant that we could radically change the speed of genome sequencing used for clinical diagnosis and the study of the difficult regions of the human genome. One could only imagine taking on and succeeding in resolving both *de novo* assembly and variant calling problems for nanopore sequencing. For this reason, it has been highly satisfying to be a part of the story of the maturation of nanopore sequencing. My work played a crucial role in showing accurate *de novo* assembly of a large cohort and taking the variant caller I helped develop in a critical care setting to diagnose 13 critically ill patients with nanopore sequencing.

This dissertation details my primary contributions to analysis methods for nanopore long-read sequencing. Most of the chapters in this dissertation correspond to a published work and can be read independently. Chapter 2 provides an overview of DNA sequencing technologies, genome inference techniques and applications of deep neural networks for nanopore sequencing technology. Chapter 3 describes the nanopore variant caller PEPPER-Margin-DeepVariant I developed that outperformed short-reads in identifying single nucleotide polymorphisms when using a linear reference. Chapter 4 describes a *de novo* genome assembler Shasta and an assembly polisher HELEN that I developed to generate contiguous and accurate assemblies with highly multiplexed promethION sequencing device. Chapter 5 describes my contributions to validate and polish the first complete telomere-to-telomere human genome assembly. Chapter 6 describes my contributions to a pipeline that uses PEPPER-Margin-DeepVariant to diagnose 12 critically ill patients with a 42% positive diagnosis rate. Finally, chapter 7 concludes this work by discussing current state of long-read sequencing and future directions.

Part II

Background

Chapter 2

Background

Providing a background in genome analysis and deep learning requires drawing attention to many biotechnology and computer science milestones. Deep learning is so widely used in bioinformatics that a critical review would deter our attention from the core topic of applying deep learning to improve human genome analysis with nanopore sequencing. To restrict my attention to the relevant topics, I provide a background of DNA sequencing technologies, genome inference methods and deep learning applications in nanopore sequencing for human genomics in this chapter.

DNA sequencing technology

Sequencing technologies aim to reconstruct the arrangement of nucleic acid within a DNA, which contains all genetic instructions for an organism to function. The application of DNA sequencing can range from the individual prognosis of disease to uncovering the history through the lens of population genetics. The ever-improving sequencing technologies provide modern biologists a versatile tool to investigate life and doctors to provide personalized care. Today, several platforms can sequence a molecule of interest. The difference between platforms ranges in sequencing read length to accuracy and applicability. In this section, I will describe commonly used DNA sequencing technologies.

Sanger sequencing

The first solution to DNA sequencing came from Fredrick Sanger in the year 1977, which is famously named *Sanger Sequencing*. Fredrick Sanger used electrophoresis and radiolabeling to identify chain-terminating synthesized DNA molecules[185]. Later fluorescent chain terminators replaced radiolabeling, and capillary electrophoresis replaced electrophoresis, which resulted in a sequencing technology that can produce reads up to 1000 bp that are highly accurate[199]. Although this technology is highly accurate (99.999%) and cost-effective for small-scale sequencing, the throughput bottleneck prohibits tackling most large-scale problems in human genomics. However, Sanger Sequencing is used for orthogonal confirmation of genetic variants and in many small-scale projects.

Next-generation sequencing technology

The successor of Sanger Sequencing is next-generation sequencing (NGS) technology. Although NGS used to mean an array of sequencing technologies, the vast popularity of

Illumina made the phrase *NGS* synonymous with Illumina sequencing technology [134]. Illumina sequencing technology uses a modified polymerase chain reaction (PCR) to generate identical copies of a DNA molecule in a cluster using a modified PCR. The clusters are then read one base at a time and terminated with a fluorophore-labeled nucleotide which is then detected optically by its fluorescence. Illumina sequencing technology is massively parallel and can generate millions of short reads (50-400bp) from a single sample. As the reads generated by this technology are generally short in length, this technology is often referred to as short-read sequencing technology [13].

Third-generation sequencing technology

Short-read sequencing technology provides a cost-effective way to study the genome, but short-reads have a difficult time in aligning to the repetitive and difficult regions of the genome[153]. In recent years, a new sequencing technology known as third-generation long-read sequencing technology has emerged to be useful as long-reads can map more accurately in difficult regions, including highly-repetitive centromeres[136]. In contrast to 50bp-400bp reads with short-read sequencing, long-read sequencing can generate reads in the range of 10kb to several Mb in length directly from native DNA [89, 90, 223, 194]. Two popular sequencing platforms, Pacific Biosciences (PacBio) and Oxford Nanopore Technology (ONT), provide a range of long-read sequencing devices. For all long-read sequencing platforms, having longer reads over short reads come at the cost of accuracy [118].

PacBio sequencing technology

PacBio is a Single Molecule, Real-Time (SMRT) sequencing platform that can generate reads in the length of kilobases. PacBio SMRT sequencing starts by creating a template known as *SMRTbell*. *SMRTbell* is a circular DNA molecule constructed with a DNA insert with single-stranded hairpin adapters on either end. The length of the DNA insert can range from hundreds of bases to several kilobases. Once the circular *SMRTbell* templates are generated, it is loaded onto a *SMRTCell* which contains millions of zero-mode waveguides to perform the sequencing. The sequencing process starts with a polymerase process to incorporate fluorescence labels into the strand. During the fluorescence incorporation, the emitted fluorophore gets recorded. This process is repeated to decode the sequence of the DNA template[118]. The first read data type introduced by PacBio is known as Continuous Long Reads (CLR), which can produce much longer (60kb - 100kb) but erroneous reads with read accuracy around 88% to 90%[178, 118].

PacBio subsequently introduced HiFi reads [221] which are an improvement over its previous continuous long reads (CLR). In HiFi mode, the DNA insert is limited to 10kb-25kb, which allows the polymerase to read the template multiple times, generating multiple subreads for each DNA template. The subreads of each template are assembled to create a consensus sequence using the PBCCS algorithm. PBS first creates a partial order alignment (POA) of the subreads and then uses a hidden Markov model to generate the highly accurate consensus sequence known as circular consensus sequence (CCS) [7]. PacBio HiFi sequencing can generate reads that are 99% accurate using this circular consensus approach [118]. Higher accuracy CCS reads require a higher number of passes of each template [7, 101] which subsequently results in a much longer run-time [118]. Recently, a new method called DeepConsensus [7] showed the read quality can be further improved by applying deep neural network-based inference to further polish the CCS sequences.

Oxford nanopore sequencing technology

Oxford Nanopore Technology (ONT) is a long-read sequencing platform that passes the DNA molecule through a pore embedded in a membrane. An electric potential is applied to the membrane, resulting in a signal encoded as current flow as the DNA transits through the pore. Different DNA bases create distinct current signals that can be used to decode the sequence going through the pore [174]. The process of decoding bases from signals is known as basecalling [174, 118, 194].

ONT sequencing starts by attaching a sequencing adapter to the double-stranded linear DNA molecule loaded with a motor protein. Then the resulting DNA mixture is loaded onto a flow cell containing several thousands of nanopores embedded in a membrane. The motor protein helps the negatively-charged DNA molecule pass through the pore using an electric current. As the DNA bases go through the pore, it disrupts the current to create unique signals that can infer the sequence of the DNA molecule. ONT provides the flexibility to generate much longer reads where 100kb+ reads are common [194] with megabase-length reads are often reported [135].

Oxford nanopore sequencing technology provides a scalable solution to long-read sequencing. However, the error rate of this sequencing platform is much higher compared to PacBio HiFi data type[118]. The majority of the errors in nanopore reads happen in homopolymer regions where a single base is repeated multiple times [174, 193]. The current signal in homopolymer regions makes it difficult to distinguish how many bases passed through the pore. However, recent advancements in basecalling algorithms and flowcell updates show continuous improvement with this sequencing platform [190, 225].

Genome inference

DNA sequencing of a sample produces millions of reads representing the fragments of the DNA molecule. As most of the reads are usually much smaller than a chromosome, we need a process to infer the sample sequence using the read fragments. The process of deriving a sample's genome using data from DNA sequencing is known as genome inference. Genome inference methods depend heavily on sequencing platforms and the characteristics of sequencing technology. Depending on the type of inference required for genome inference, the methods can be broken into two parts: *de novo* assembly where a reference genome is not used and variant calling where the sequence is compared against a known or established reference genome.

De novo assembly

de novo assembly of a genome determines a sample's genome using the reads and their overlaps without using a reference genome. Generally, the process starts with finding overlaps between the reads and creating a genome graph, and then untangling the graph to reveal the linear sequence that represents the sequence of the sample [26, 194, 149, 148]. Accurate genome assembly is challenging as the reads are generally shorter than whole chromosomes.

The assembly methods are specific to sequencing platforms to model their error characteristics and take advantage of the read lengths. The process of finding overlaps between reads needs to consider sequencing errors that are platform-specific [118, 174, 128]. Once the overlaps between reads are sufficiently determined to occur due to reads originating from the same region of the genome and not due to sequencing errors, they can be represented as a graph.

The most common sequence graphs used in genome assembly are overlap graphs and de Bruijn graphs. A vertex in an overlap graph corresponds to a sequencing read, and edges are added between reads when sufficient evidence of a match between reads. These graphs can then be used to generate a consensus sequence to produce an assembly [142]. In de Bruijn graphs, the sequence of each node is a *k-mer* found on a read. Edges represent exact match between two k-mers between reads [165, 175]. Overall, de Bruijn graphs contain much less information than overlap graphs but benefit from being more efficient.

Genome inference through *de novo* assembly aims to achieve the highly contiguous and accurate reconstruction of the genome. The sequencing data type used for genome assembly can directly affect the continuity and quality of the genome assembly. Short-read sequencing technology has mainly been problematic in generating contiguous genome assemblies[98, 82, 23] as it is challenging to find overlaps between short-reads in repetitive regions[198, 90, 135].

The telomere-to-telomere (T2T) consortium recently published the first complete sequence of a human genome[148]. The construction of the gapless complete human genome was done with two contemporary long-read sequencing platforms, PacBio-HiFi and ONT[148]. Although short-reads were used to validate and remove platform-specific errors from the assembly[128], it was the long-read sequencing platforms that made the primary assembly possible. The consortium also shed light on the centromere [5] and how the complete genome assembly improved the assessment of human genetic variation [2]. In all of these studies, long-reads played a central role in showing that longer and more accurate reads enable the study of the most repetitive regions of the genome.

Variant calling

Variant calling is the process of deriving the difference between a sample's genome sequence and a reference sequence. Variant calling starts by mapping the reads to a reference genome that closely represents the sample sequence. Then using the alignment, computational methods determine the differences between the reference and the sample in the form of a set of variants. The advantages of using reference-based variant calling can be the utilization of heavily studied and recorded information about the reference sequence and a reasonably straightforward way to determine the variation. However, reference-based methods are known to introduce bias which is exacerbated when the sample of interest has different genomic sequence construction due to ethnicity[198].

Genomic variation of a sample compared to a reference can range from small variations 1bp to 50bp including single nucleotide polymorphisms (SNPs) or insertions and deletions (INDELs), to large scale (50bp+) Structural Variatoins (SVs). Variant calling methods are generally classified between small variant calling and structural variant calling methods. There is a general agreement that SVs require different treatment and algorithm design compared to small variant calling methods [193, 153, 216].

Sequencing technology and the error rate of the sequencing data have direct implications in deriving accurate variant calls. Short-reads can produce precise, highly small variants [153] in mappable regions of the genome but fails to derive structural variants accurately[187] as short-reads do not span through the variant. Highly accurate short-reads can generate precise variant calls but have difficulty mapping to repetitive regions like segmental duplications or MHC regions. Recently, a community challenge called precisionFDA v2 revealed that variant calling with long-reads has the advantage over short reads while using a linear reference sequence as long-reads map accurately to the most repetitive regions of the genome[153].

Although long-reads are desirable for deriving small variants, the high error rate of Oxford Nanopore reads makes it difficult to call variants accurately. SNP calling with ONT is highly accurate, but INDEL calling suffers significantly due to the non-random errors in the sequencing platform. PacBio-HiFi, on the other hand, is highly precise and recent surveys show that HiFi is the most suitable data type to derive small variant calls [153, 216, 193].

Computational methods for variant calling generally use a probabilistic model curated for the sequencing platform to tell true variants apart from sequencing errors. Traditional methods like GATK [130] or freebayes [64] used statistical models that are tuned for Illumina short-read sequencing technology. As long-read sequencing technology came along, tuning parameters for the statistical models became increasingly difficult due to the error characteristic between sequencing platforms being vastly different [174]. Later the DeepVariant [166] variant caller used a deep neural network-based approach that can be simply re-trained for different sequencing platforms to call variants.

Deep learning for genome inference

Genomics aims to unravel the functional utility of genomic elements through various technologies, including genome sequencing [81]. The field of genomics is centered around data that captures the properties of the biological element of interest[18, 71]. In genome inference, we aim to discover novel properties of a sample's genome, like genomic variation [33]. However, genomics data can be too large or complex to simply do a visual inspection [51, 174]. The human genome is 3.3 GigaBases, making it impossible to visually inspect every base.

Similarly, technologies that we use to generate genomic data can directly affect the inference process [174, 193]. For example, studying the genome with short reads vs. long reads would pose different challenges. Short reads will have difficulty mapping to repetitive regions [153], whereas long reads have higher sequencing errors than short reads [174]. The genomics community depends on statistical models to assist in automating the genome inference process [162]. The statistical or machine learning models require features to capture patterns in the data [141]. Thus these tools are often platform-specific, and designing these tools requires domain expertise to create handcrafted features [141, 51].

Unlike traditional algorithms where hypothesis about the data is hardcoded or assumed, deep learning algorithms are designed to find patterns in data automatically [73, 103]. Deep learning is a discipline of machine learning that proposes deep neural networks which perform successive operations to compute increasingly complex features in the data [141]. Deep neural networks are particularly well suited for genomics as they can improve inference quality by revealing patterns in extensive genomic data where finding hypotheses based on the spatial or morphological pattern is difficult [141, 51, 21].

Applications of deep neural network in nanopore sequencing

Nanopore sequencing measures DNA or RNA molecules flowing through the pore as current signals and this method is fundamentally different from the traditional sequencing-by-synthesis approach[39]. The sequencing process then converts current signals to the nucleotide sequence. This conversion of the signal to sequence is an inference problem known as basecalling that needs statistical modeling based on the features observed in the signal. The first generation of nanopore basecallers like Nanocall used Hidden Markov Models (HMMs) for basecalling [36]. These HMM-based models used k-mer emission probabilities to predict the observed sequence [36, 174]. However, the sequencing device's translocation speed is not constant, and encoding such features became increasingly difficult [174]. Rather than using hand-crafted attributes, Albacore and Guppy introduced Recurrent Neural Networks for basecalling [225]. Recurrent Neural Networks are most effective in sequence translation problems as they can find abstraction in sequential data [132]. A recent iteration of the basecaller named Bonito uses convolutional layers on top of a Connectionist Temporal Classification (CTC) decoder [76] and currently known to have the highest accuracy [225]. The introduction of deep neural networks in basecalling and additional chemistry improvements boosted the intial sequence accuracy of nanopore from 60% to current state-of-the-art 98.2% in over six years [118].

Nanopore reads usually encounter INDEL-like errors in homopolymer regions [174, 118, 225, 224]. As a result of the reads being erroneous, genome assemblers that do not perform any read correction generally produce consensus sequence that has small INDEL-like errors which creates issues in downstream analysis like gene annotation [74, 128]. These errors can generally be eliminated using consensus correction methods or polishers [209]. Initial methods used high-quality short-reads from same sample to correct these errors [218]. Current improvements in basecallers and polishers that use deep neural networks can produce highly-accurate (99.9%) consensus sequences using nanopore reads only [128, 194].

The utility of nanopore goes beyond traditional genome sequencing as it can sequence full-length RNA molecules and detect methylation or modification in nucleotide bases. Deep Neural Networks (DNNs) have proven to be driving factors of these applications. DNNs have been applied to detect DNA base modifications [146, 117] and differential modification from direct RNA sequencing with nanopore [169]. As nanopore long reads can span most difficult regions of the genome, these application made it possible to study the epigenetic regulation in repeat-rich regions [135].

Part III

Haplotype-aware variant calling

Chapter 3

PEPPER-Margin-DeepVariant: Haplotype-aware variant calling pipeline for long reads.

Preamble

This chapter contains the text from "Haplotype-aware variant calling with PEPPER-Margin-DeepVariant enables high accuracy in nanopore long-reads," published in Nature Methods in November 2021. The manuscript details a long-read variant calling pipeline PEPPER-Margin-DeepVariant that can utilize the haplotype information of the sample for genotyping. The described method won two awards in precisionFDA challenge v2 in Oxford Nanopore category.

I share the first co-authorship of this manuscript with Trevor Pesout who developed Margin and Pi-Chuan Chang who enabled DeepVariant to work on nanopore reads. My contribution in this manuscript is designing the pipeline, developing PEPPER and deploying the pipeline for general usage. I want to thank Maria Nattestad, Alexey Kolesnikov, Sidharth Goel, Gunjan Baid, Mikhail Kolmogorov, Jordan Eizenga, Karen Miga, Paolo Carnevali, Miten Jain, Andrew Carroll, Benedict Paten, who are the co-authors of this manuscript. I consider this collaborative work between Google genomics and the UCSC genomics institute as the highlight of my graduate research.

Introduction

Most existing reference-based small variant genotyping methods are tuned to work with short-reads [34, 130]. Short-reads have high base-level accuracy but frequently fail to align unambiguously in repetitive regions [115]. Short-reads are also generally unable to provide substantial read-based phasing information, and therefore require using haplotype panels for phasing [24] that provide limited phasing information for rarer variants.

Third-generation sequencing technologies, like linked-reads [10, 53, 220] and long-reads [87, 50], produce sequences that can map more confidently in the repetitive regions of the genome [85], overcoming the fundamental limitations of short-reads. Long-reads can generate highly contiguous *de novo* assemblies [135, 120, 194, 26, 149, 92, 183], and they are increasingly being used by reference-based analysis methods [123, 125, 48, 221, 46, 82, 187, 163]. The Genome-In-A-Bottle Consortium (GIAB) [232] used the additional power of long-reads and linked-reads to expand the small variant benchmarking set to cover more of the genome [216]. 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[153].

Oxford Nanopore Technologies (ONT) is a commercial nanopore-based high-throughput [194] long-read sequencing platform that can generate 100kb+ long reads [194, 88]. Nanopore long-reads can confidently map to repetitive regions of the genome [85] including centromeric satellites, acrocentric short arms, and segmental duplications [135, 90, 56, 49]. Nanopore sequencing platform promises same-day sequencing and analysis [52], but the base-level error characteristics of the nanopore-reads, being both generally higher and systematic, make small variant identification challenging [174].

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 [221]. The overall accuracy of PacBio-HiFi-based variant identification is competitive with short-read based methods [221]. These highly accurate long-reads enabled the small variant benchmarking of major histocompatibility complex (MHC) region [30] and difficult-to-map regions [216].

In our previous work, we introduced DeepVariant, a universal small variant calling method based on a deep convolutional neural network (CNN) [166]. We showed that by retraining the neural network of DeepVariant, we can generate highly accurate variant calls for various sequencing platforms [166]. 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 [194, 174] 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 [24, 221, 82, 163, 47, 180, 29, 95, 168]. Previously, we trained DeepVariant on PacBio HiFi long-read data and showed highly competitive performance against short-read based methods for small variant identification [221]. 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[123], Clair [125], and longshot [48]. 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 [79] 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.

Results

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 3.1a). 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 [166], 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 [166, 221] 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 A.2).

Existing variant callers, like Clair[125], 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[46, 221]. The schema of the PEPPER-Margin-DeepVariant pipeline follows a similar design of PacBio HiFi-based DeepVariant[221] and Medaka[123] that use haplotyping to provide better genotyping results. However, Medaka[123] 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 A.2).

We trained PEPPER-Margin-DeepVariant on HG002 sample using Genome-In-A-Bottle (GIAB) v4.2.1 benchmarking set [216]. We trained PEPPER and DeepVariant on `chr1-chr19` and tested on `chr20` and used `chr21-chr22` as holdout sets (See Online Methods).

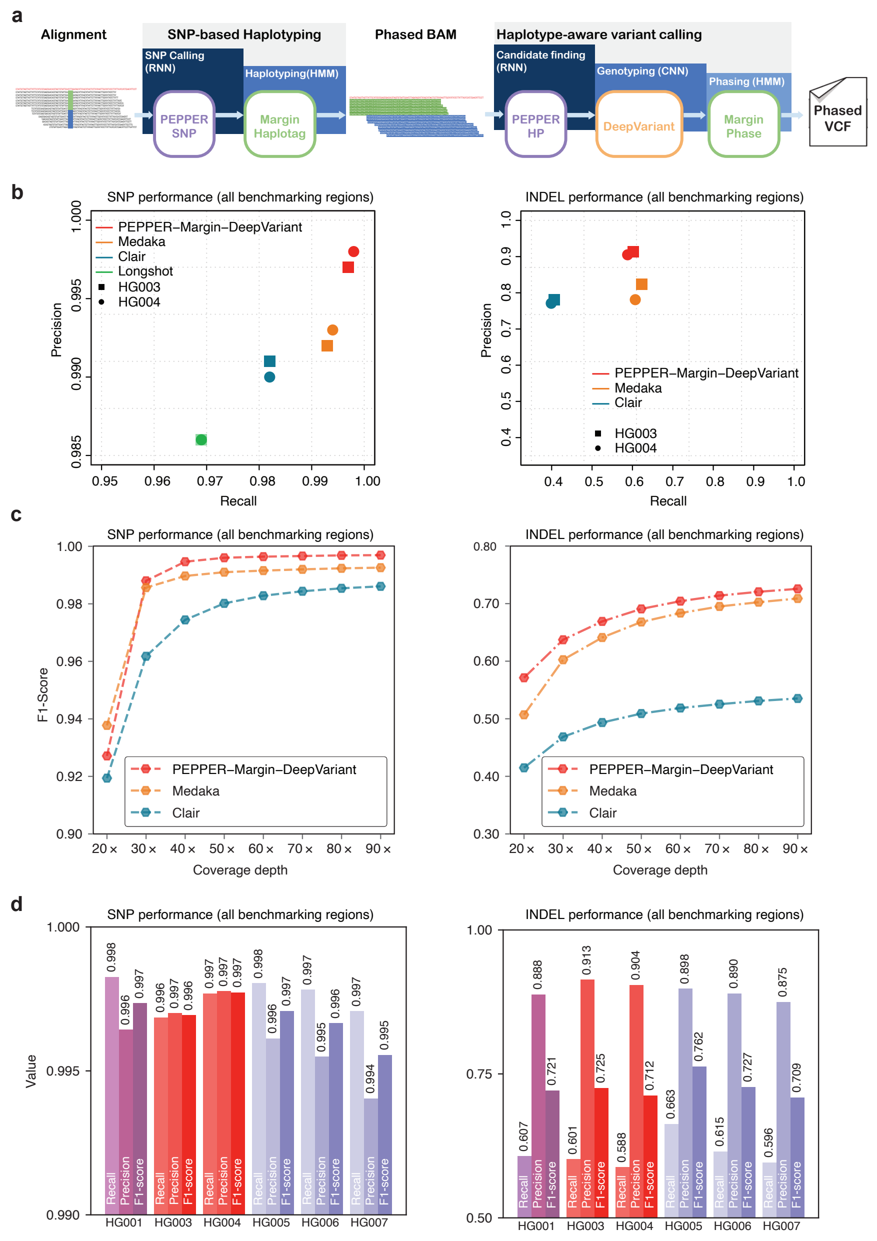


Figure 3.1: **Nanopore variant calling results.** **(a)** Illustration of haplotype-aware variant calling using PEPPER-Margin-DeepVariant. **(b)** Nanopore variant calling performance comparison between different nanopore-based variant callers. **(c)** Evaluating variant calling performance at different coverage of HG003. **(d)** Variant calling performance of PEPPER-Margin-DeepVariant on six GIAB samples.

Nanopore variant calling performance

We compared the nanopore variant calling performance of PEPPER-Margin-DeepVariant against Medaka[123], Clair[125], and Longshot[48]. 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 3.1b, Supplementary table A.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 [236] and v4.2.1 [216]). 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 A.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 A.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 3.1c, Supplementary table A.4). At coverage above 30×, PEPPER-Margin-DeepVariant achieves a higher F1-score than Medaka and Clair (Supplementary table A.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 A.6) and against GRCh37 and GRCh38 reference genomes (Supplementary Table A.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 3.1d, Supplementary table A.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 A.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 [221, 166]. We analyzed the PacBio HiFi PEPPER-Margin-DeepVariant variant calling performance on the 35x HG003 and HG004 from precisionFDA [153] against DeepVariant-WhatsHap-DeepVariant (current state-of-the-art method [153, 221]) 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 A.10).

We compared the run-time and cost of Oxford nanopore-based variant calling pipelines on 50× and 75× HG001 data (Supplementary table A.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 3.1, Supplementary Table A.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 A.12, Supplementary table A.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.

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 [8]. 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 [153]. 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.

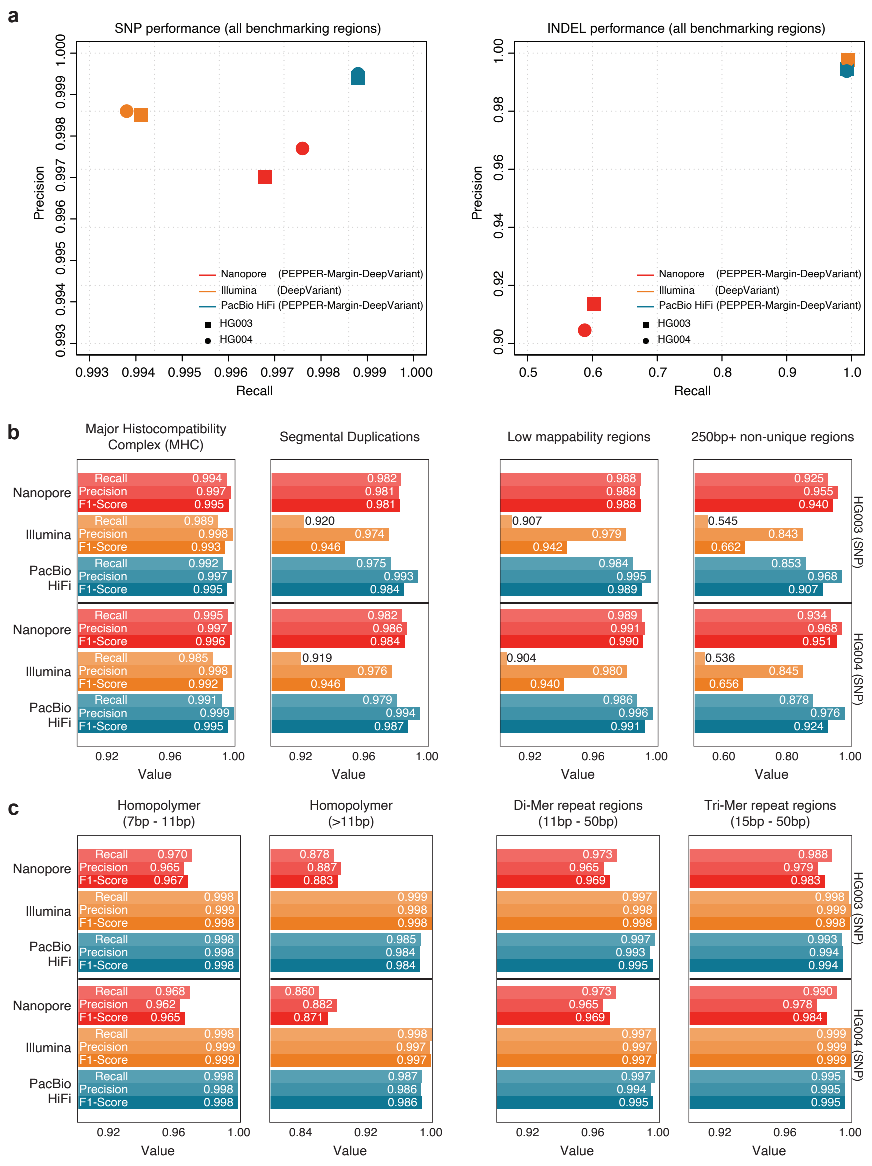


Figure 3.2: **Comparison between Nanopore, Illumina and PacBio HiFi variant calling performance.** **(a)** SNP and INDEL performance comparison of Nanopore, Illumina and PacBio HiFi in all benchmarking regions. **(b)** SNP performance comparison in difficult-to-map regions of the genome. **(c)** SNP 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 A.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 3.2a, Supplementary table A.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 3.2b, Supplementary table A.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 3.2c, Supplementary table A.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 [216]. 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 A.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 A.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.

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.

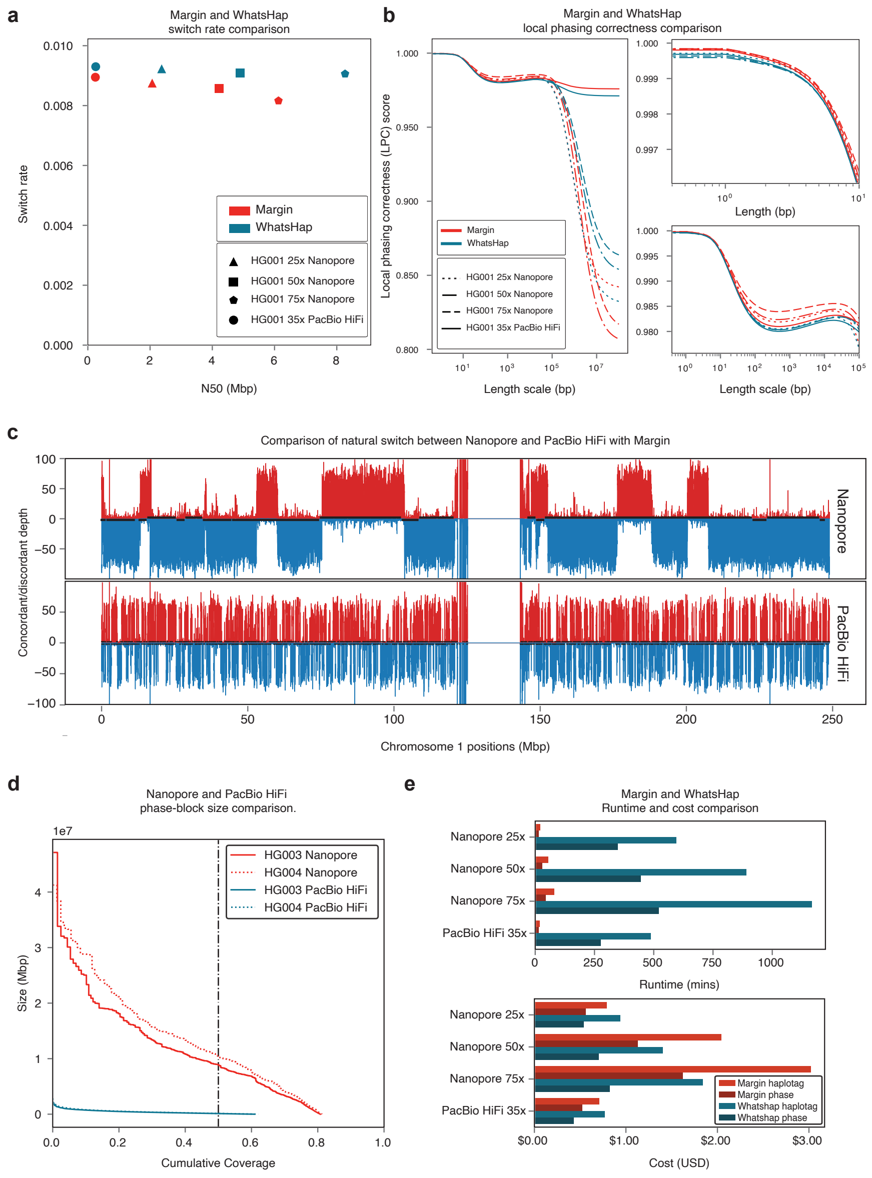


Figure 3.3: **Margin and WhatsHap phasing results. (a)** Phaseset switch rate to N50. **(b)** Novel metric "Local Phasing Correctness" analyzing phaseset accuracy across different length scales. **(c)** "Natural Switch Rate" describing haplotagging accuracy for reads. **(d)** Phaseset N50 for Nanopore 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 3.3a, Supplementary Tables A.20 and A.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 3.3b). 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 3.3c, 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 A.22). Full plots including local haplotagging accuracy visualization are shown in Supplementary Figures A.3, A.4, A.5, A.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 A.23, Supplementary Table A.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 A.24)

Gene Analysis

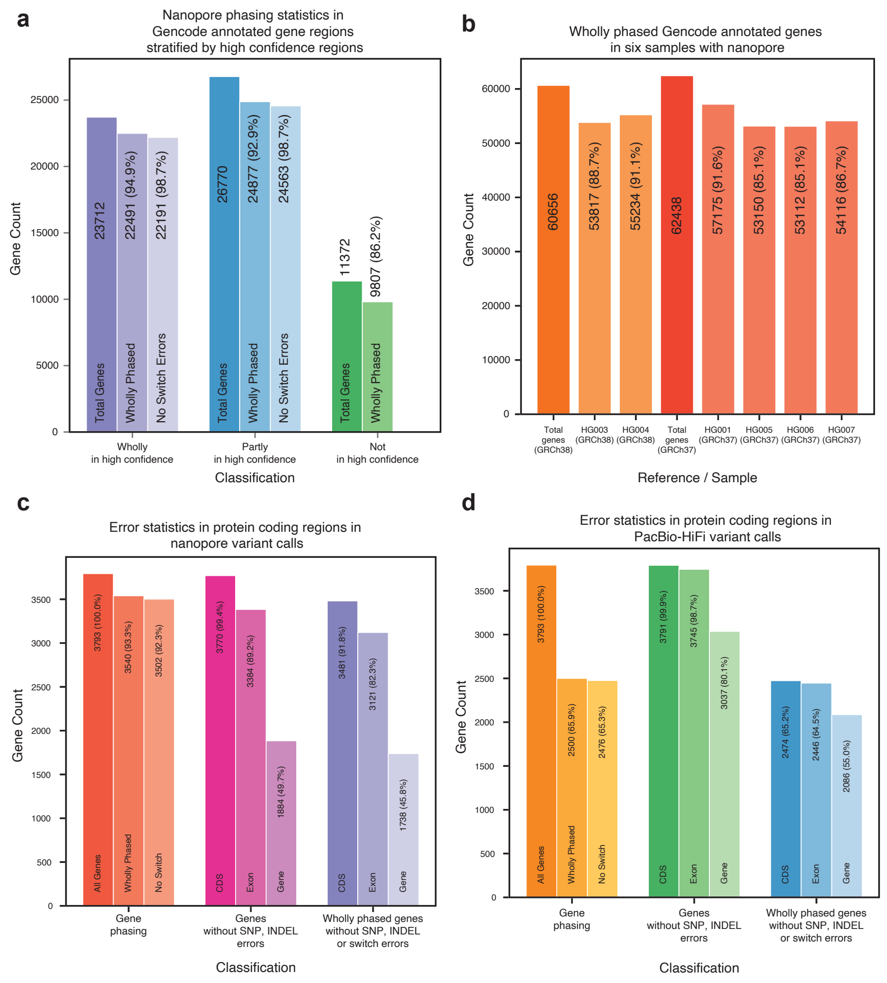


Figure 3.4: **Gene analysis.** **(a)** Phasing analysis for HG001 over GENCODE annotated gene regions, stratified by GIAB high confidence coverage. Percentages are relative to their predecessor. **(b)** Wholly phased GENCODE annotated gene regions. Percentages are relative to the total genes annotated 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[60] (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 3.4a, 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 3.4b, 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 A.25, A.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 3.4c) and 35x PacBio HiFi (Figure 3.4d) reads (Supplementary Tables A.27, A.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.

Diploid polishing of *de novo* assemblies

Oxford Nanopore-based assemblers like Flye [92] and Shasta [194] 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 3.5a). It can polish haploid Oxford Nanopore-based assemblies with either Nanopore or PacBio HiFi reads.

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.

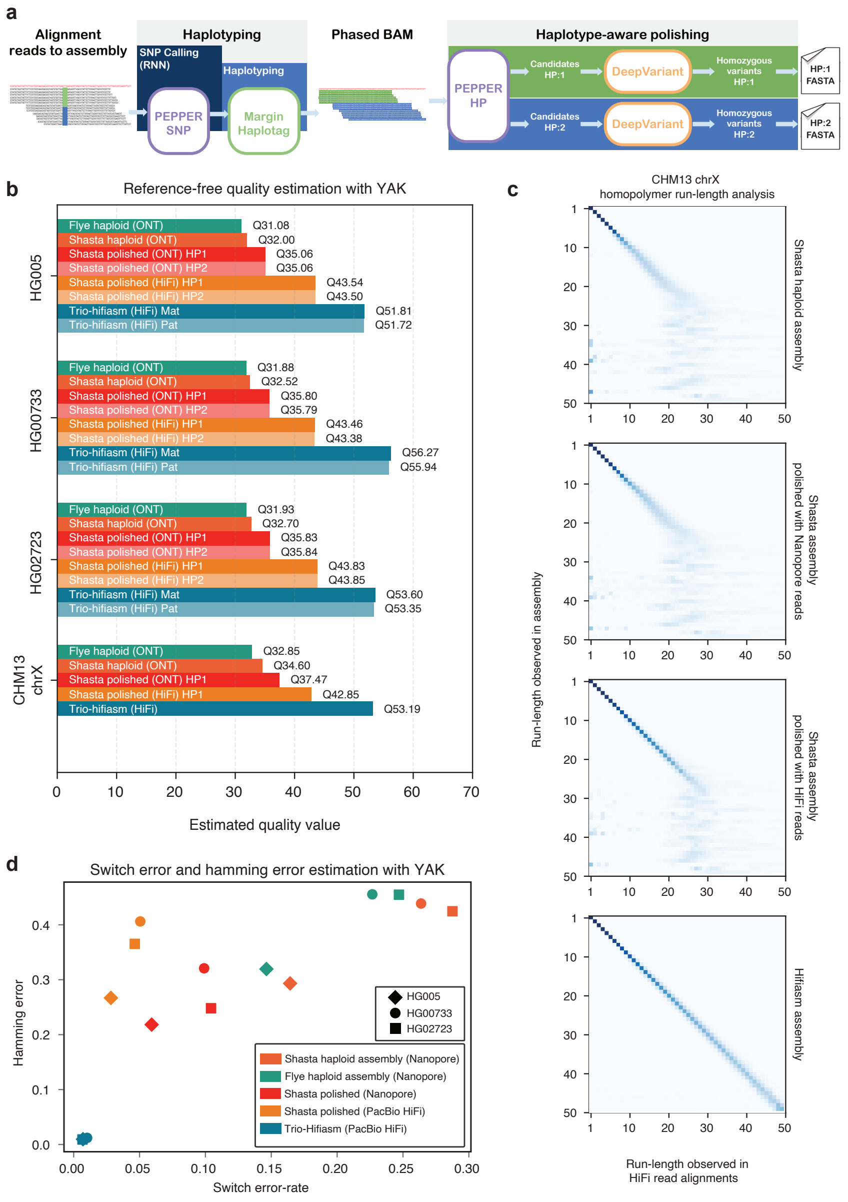


Figure 3.5: **Diploid assembly polishing results. (a)** Illustration of the diploid assembly polishing pipeline. **(b)** Estimated quality values of assemblies using YAK. **(c)** CHM13-chrX run-length confusion matrix between different assemblies and PacBio HiFi reads aligned to the corresponding assembly. **(d)** Switch error and hamming error comparison between assemblies.

Diploid *de novo* assembly polishing performance

We generated haploid assemblies using Shasta [194] and Flye [92] 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[26], 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[26]. 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 A.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 3.5b, Supplementary table A.29, Supplementary table A.30).

The dominant error modality for ONT data are homopolymers[194]. In Figure 3.5c 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.

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 3.1: 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[112] 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.

Figure 3.5d 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 3.5d, Supplementary table A.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 3.1, 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 [80] and then validated the reconstructed SV sets using the previously described approach [233]. 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 A.31).

Discussion

Long-read sequencing technology is allowing gapless human genome assembly [135] and enabling investigations in the most repetitive regions of the genome[153].

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 [194]. 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 [204, 19] and clinical genomics [69]. 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.

Methods

Analysis methods and data pre-processing

Read alignment

We used `minimap2` [110] version `2.17-r941` and `pbmm2` version `1.4.0` to align reads to a reference genome. The supplementary notes have details on execution parameters.

Subsampling alignment files to different coverages

We used `samtools` [113] version `1.10` to generate alignment files of different coverages. The supplementary notes have details on execution parameters.

Variant calling

We used the following methods to call variants with nanopore data:

- `PEPPER-Margin-DeepVariant` version `r0.4`.
- `Medaka` [123] version `v1.2.1`.
- `Clair` [125] version `v2.1.1`.
- `Longshot` [48] version `v0.4.2`.

For Illumina short-reads and PacBio HiFi we used `DeepVariant` version `v1.1.0`. The details on execution parameters are available in supplementary notes.

Benchmarking variant calls

We used `hap.py` [99] version `v0.3.12` to assess the variant calls against GIAB truth set. The `hap.py` program is available via `jmcdani20/hap.py:v0.3.12` docker image. The command used for the assessment is described in the supplementary note.

For HG002, HG003, HG004, HG005 we used GIAB v4.2.1 truth set [216] against `GRCh38` reference and for HG001, HG005, HG006, HG007 samples, we used v3.3.2 variant benchmarks [236] against `GRCh37` reference genome. We used GIAB stratification `v2.0` files with `hap.py` to derive stratified variant calling results. The GIAB benchmarking data availability is listed in data availability section of supplementary notes.

Mendelian Analysis

For our Mendelian Analysis we used RTG version 3.12 [32].

Phasing and haplotagging

We used `Margin` version `v2.0` and `WhatsHap` [46] version `v1.0` to haplotag and phase the variants. `Margin` is available in `https://github.com/UCSC-nanopore-cgl/margin` and `WhatsHap` is available in `https://github.com/whatshap/whatshap`. The details on how we ran these tools is described in supplementary notes.

Small variant switch error rate and hamming error rate determination

We used a Workflow Description Language (WDL)-based analysis pipeline `whatshap.wdl` available in `https://github.com/tpesout/genomics_scripts` to derive the switch error rate and hamming error rate compared to the GIAB truth set. The `whatshap.wdl` workflow invokes the `stats` and `compare` submodules available in `whatshap` version `v1.0`.

In our analysis, we compared phased variants against GRCh37 reference against GIAB v3.3.2 truth set to derive switch error rate and Hamming error rate. We only considered variants that have `PATMAT` annotation in the truth set and that fall in the high-confidence region defined by GIAB benchmarking set. The non-`PATMAT` annotated variants in the GIAB benchmarking variant set are not trio-confirmed so we did not use those to benchmarking our phasing methods. We used `whatshap compare` command to generate the whole genome switch error rate and used a custom script defined in `whatshap.wdl` to derive the hamming error rates.

Local phasing correctness calculation

For the Local Phasing Correctness (LPC) analysis, we used the `calcLocalPhasingCorrectness` executable found in the `https://github.com/UCSC-nanopore-cgl/margin` repository. The LPC analysis require a truth variant set and a query variant set. We used GIAB benchmarking set as truth. The `calcLocalPhasingCorrectness` generates a `tsv` file describing the results.

We used

`https://github.com/tpesout/genomics_scripts/plot_haplotagging_lpc.py`

script to visualize the results. The details of parameters is described in supplementary notes. The methods used in local phasing correctness as a metric is presented separately in the methods description.

Haplotagging accuracy and natural switch determination

We used

`https://github.com/tpesout/genomics_scripts/haplotagging_stats.py`

to calculate the haplotagging accuracy. The script calculates average haplotagging accuracy and average tagged reads per 10kb. Details on execution parameters is available in supplementary notes.

We visualized the natural switch error using `compare_read_phasing_hapBam.py` available in `https://github.com/tpesout/genomics_scripts`. Details on execution parameters is available in supplementary notes.

Phaseblock N50 calculation

An N50 value is a weighted median; it is the length of the sequence in a set for which all sequences of that length or greater sum to 50% of the set's total size. We used the `ngx_plot.py` available from

`https://github.com/rlorigro/nanopore_assembly_and_polishing_assessment/`

to plot phaseblock N50. From a phased VCF file, we extracted the phaseblock name, contig, start position, and end position to create the input file. We used `3272116950` as the size of the genome to maintain consistency with previous work [194].

Variant calling and phasing analysis on Gencode annotated regions

We used Gencode v35 [79] to determine the variant calling and phasing accuracy in gene regions. The Gencode data is publicly available and can be found in data availably section of supplementary notes. We used `https://github.com/tpesout/genomics_scripts/gencode_to_stratification.py` script to convert the Gencode regions to a bed file that is acceptable to `hap.py`. With the newly defined stratified regions from Gencode, we ran `hap.py` to determine the variant calling accuracy in gene regions.

Nanopore and PacBio HiFi *de novo* assembly generation

We used `Shasta` [194] with a development build after version `0.7.0` available from `https://github.com/chanzuckerberg/shasta` (commit `06a639d36d26a4203c0b934d6e63c719750c5398`) and `Flye` [92] version `2.8.2` available from `https://github.com/fenderglass/Flye` to generate nanopore-based *de novo* assemblies. For PacBio-HiFi-based assemblies we used `hifiasm` [26] version `0.14` available from `https://github.com/chhylp123/hifiasm`. The commands used to generate the assemblies are provided in execution parameters section of supplementary notes.

Assembly QV and switch error rate analysis

We assessed the assemblies with `yak` [26] version `0.1` available from `https://github.com/lh3/yak`. YAK is a short-read kmer-based assembly quality estimator. We use short-reads for each sample to estimate the quality of the assembly with *k-mer* size of 31, With parental short-reads, YAK can also estimate the switch error rate in the assembly. WDL version of the pipeline `standard_qc_haploid.wdl` is available in `https://github.com/human-pangenomics/hpp_production_workflows/`.

Homopolymer run-length analysis

We used `runLengthMatrix` module of `margin` to derive the homopolymer run-length analysis between assembly and reads. In `runLengthMatrix`, we convert each read sequence into RLE form and track a map of raw positions to RLE positions. We convert from a raw alignment to RLE alignment by iterating through the matches in raw space and tracking the previous RLE match indices. From this set of matched read and reference RLE positions, we construct a confusion matrix. Details of the command are provided in execution parameters section of supplementary notes.

Small variant accuracy evaluation of assemblies

We used `dipcall` [112] to identify the small variants from the assemblies. The `dipcall` variant identification takes the maternal and paternal haplotypes generated by a phased assembly and a reference genome sequence. It maps the haplotypes to the reference and generates a VCF file containing all small variants identified in the assembly. For the haploid assembly, we provided the haploid assembly as both maternal and paternal haplotypes to `dipcall`. `dipcall` also generates a bed file containing regions where the assembly maps to the reference. We intersected the bed files to get regions that are assembled by all assembly methods and intersected with GIAB high-confidence region. Finally, we used `hap.py` to compare the variant calls derived from the assemblies against GIAB benchmarking VCF to get the accuracy statistics. Please see supplementary notes for `dipcall` parameters.

Structural variant accuracy evaluation of assemblies

We evaluated SV precision and recall for each assembly as follows. We aligned each set of contigs (either haploid or diploid) against the reference with minimap2 v2.18 using default parameters and the "asm5" preset. We have selected the hg19 reference (instead of the hg38 version) to be able to compare against the curated set of SVs in the HG002 genome [233] that was initially produced using the hg19 reference. Given the reference alignments, we used `svim-asm` 1.0.2 [80] in the respective (haploid or diploid) mode to call SV of size more than 50bp. Given two sets of SV calls, we used the SVbenchmark tool from the SVanalyzer package v0.36 with default parameters to estimate recall and precision. To estimate recall for homozygous and heterozygous SVs separately, we split each truth set into two respective parts. The statistics were evaluated within the GIAB Tier1 high-confidence regions that cover 2.51Gb of the human genome [233].

The assemblies produced by hifiasm had high recall (97.8% homozygous 97.0% heterozygous) and precision (93%) against the HG002 curated SVs set (Supplementary Table A.31). In addition, the recall and precision of the Shasta and PEPPER-Margin-DeepVariant assemblies measured against the HG002 curated SVs were highly correlated with recall and precision measured against the hifiasm SV calls. This allowed us to estimate the SV recovery accuracy of the Shasta and PEPPER-Margin-DeepVariant assemblies for the HG005, HG0073, and HG02733 genomes, for which curated sets of SVs were not available.

Method description

PEPPER

PEPPER is a recurrent neural network-based sequence prediction tool. In PEPPER, we use summary statistics derived from reads aligned to a reference to produce base probabilities for each genomic location using a neural network. We translate the position-specific base probabilities to the likelihood values of candidate variants observed from the read alignments. We propose candidate variants with likelihood value above a set threshold to DeepVariant for genotyping. Candidate pre-filtering with PEPPER ensures a balanced classification problem for DeepVariant and achieves high-quality variant calling from erroneous long-reads.

We use PEPPER in two steps in the variant calling pipeline. Initially, we use the PEPPER-SNP submodule to find single nucleotide polymorphisms (SNPs) from the initial unphased alignment file. In this setup, we tune PEPPER-SNP to have high precision so Margin can use the SNPs confidently to phase the genome. To this end, we also exclude insertions and deletions (INDELs) from the callset as they have notably worse performance for nanopore reads. Margin can then tag reads in the alignment file with predicted haplotypes.

After Margin, we use the PEPPER-HP submodule on the phased alignment file to generate haplotype-specific likelihoods for each candidate variant observed from the read alignments. In PEPPER-HP, we consider SNPs, insertions, and deletions as potential candidate variants. We propose the candidate variants with likelihood values higher than a set threshold to DeepVariant for genotyping with a more extensive convolutional neural network (CNN). We tune PEPPER-HP to achieve high-sensitivity but low-precision during candidate finding. The PEPPER-Margin-DeepVariant suite can identify small variants with high-quality from erroneous reads.

PEPPER-SNP

PEPPER-SNP is a submodule of PEPPER used to identify single nucleotide polymorphisms from reads aligned to a reference sequence. PEPPER-SNP works in three steps: image generation, inference, and candidate finding. First, we generate summary statistics from reads aligned to a reference sequence. We encode basic alignment statistics at each genomic location in an image-like tensor format. Second, we apply a recurrent neural network to predict the two most likely bases at each genomic location. Finally, we use the base predictions from each genomic location to compute the likelihoods of SNPs we observe from the reads. We filter candidate variants with a likelihood value below a set threshold to find a set of SNPs. Likelihood thresholds for PEPPER-SNP were determined by training the model on HG002 chr1-19, selecting an appropriate threshold, and verifying on chr20. The SNP set we get from PEPPER-SNP is used by Margin to phase the alignment file.

PEPPER-SNP: Image generation

In the image generation step of PEPPER-SNP, we generate summary statistics of base-level information per genomic location. The summary provides weighted observation of bases from all reads divided into nucleotide and orientation.

In PEPPER-SNP, we do not encode insert bases observed in reads to reference alignment as we only look for SNPs. We use a position value to represent a location in the reference sequence. For each genomic location, we iterate over all reads that align to that genomic location and encode ten features to encode base-level information: $\{A, C, G, T, Gap(*)\}$ divided into two read orientations: $\{forward, reverse\}$. Finally, we normalize the weights of each genomic position based on the read coverage.

Supplemental figure A.7 describes the feature encoding scheme we use in the image generation step of PEPPER-SNP. The top row of the image, annotated as REF, describes the reference base observed at each genomic location. The colors describing the bases are $\{A : Blue, C : Red, G : Green, T : Yellow, Gap(*) : White\}$. Each row after REF describes a feature; each feature encodes an observation of nucleotide bases from a forward, or a reverse strand read. We use ten features to encode base-level information: $\{A, C, G, T, Gap(*)\}$ divided into two read orientations: $\{forward, reverse\}$. For example, A_F encodes the observations of base A from forward-strand reads, and A_R encodes observations of base A from reverse-strand reads. The columns describe genomic locations to the reference sequence.

In each column, we encode each observation as weights, which we show as alpha of each base. The grey weights are zero weights. At position (23), the weight distribution of $\{C, G\}$ bases indicates a potential heterozygous variant at that position. The REF row is shown in the figure to describe the scheme; in practice, we do not encode the REF row.

In the inference step of PEPPER-SNP, we use a recurrent neural network for sequence prediction. The network architecture consists of two bidirectional gated recurrent unit (GRU) layers and a linear transformation layer. The linear transformation layer produces a prediction of two bases for each genomic location present in the summary image. To identify potential variants, we use 15 class-labels for base prediction: $\{AA, AC, AG, AT, A*, CC, CG, CT, C*, GG, GT, G*, TT, T*, **\}$. We do not use co-linear classes like CA and AC as two separate classes, as it is not possible to differentiate between these two classes from the summary observations.

From the image-generation module of PEPPER-SNP, we get summary images in 1kb chunks. We use a sliding window method during inference and chunk the 1kb segments into multiple overlapping windows of 100bp segments. We first run inference on the leftmost window and go to the next window with 50bp overlapping bases. We pass the hidden state output from the left window to the next window and keep a global inference counter to record base-predictions.

PEPPER-SNP: Inference model

Supplemental figure A.8 describes the neural network-based inference scheme. The two dotted boxes indicate two adjacent windows with overlapping sequences. The top panel shows the inference scheme. We start with the first window and produce base predictions for each genomic location present in that window; then, we slide the window to the right. We do this from left to right on the entire genomic sequence. We record the base predictions from all windows in a global counter and report them to the candidate finder to calculate candidate likelihoods.

We trained the inference model using a gradient descent method. We use adaptive moment estimation (Adam [145]) to compute gradients based on a cross-entropy loss function. The loss function is defined to calculate the prediction performance at each genomic location against a labeled set of expected base observations derived from the Genome-In-A-Bottle (GIAB) truth set. The gradient optimization attempts to minimize the loss function by tuning the parameters of the neural network.

We trained PEPPER-SNP with 100× coverage of HG002 data subsampled at different coverage values (20x-100x). We split the training sets into three sets: train, test, and holdout. We use chromosome 1 to 19 for training, chromosome 20 for testing, and we keep 21 and 22 as holdout sets. We train the models for several epochs and test after each epoch. Finally, we pick a model that performs the best on the holdout dataset.

PEPPER-SNP: Candidate finding

In the candidate finding step of PEPPER-SNP, we take the base-predictions and derive likelihoods of SNPs we observe from the read alignments. If the likelihood value of a variant is above a set threshold, we pick that allele to be a real variant.

In PEPPER-SNP, we derive a allele probability (AP) and a non-reference observation likelihood (NR) for each observed SNP. First, at each genomic location (pos) we use the prediction vector of base-classes

$\{AA, AC, AG, AT, A*, CC, CG, CT, C*, GG, GT, G*, TT, T*, **\}$ to derive two prediction vectors $V_1[pos] = [A_1, C_1, G_1, T_1, Gap(*)_1]$ and $V_2[pos] = [A_2, C_2, G_2, T_2, Gap(*)_2]$. For example, the predicted value of class $\{GT\}$ contributes to $V_1[pos][G_1]$ and $V_2[pos][T_2]$ values. Then, we iterate over all the reads to find potential SNPs by recording each read base that does not match the reference base. We calculate the likelihood of candidate base b, observed at position pos, by taking the maximum likelihood from the prediction vectors $V_1[pos]$ and $V_2[pos]$ as shown in equation 3.1 where we denote allele likelihood as AP.

$$AP[b, pos] = \max(V_1[pos][b], V_2[pos][b]) \tag{3.1}$$

We also calculate the likelihood of non-reference base observation (NR) to estimate the likelihood of observing any allele other than the reference allele at a location. We derive non-reference base observation likelihood from prediction vectors V_1 and V_2 independently and take the maximum value between two values. For each prediction vector, we take the sum of the values, subtract the observation likelihood of the reference base, and divide by the sum of the prediction vector. Equation 3.2 describes the calculation NR in PEPPER-SNP.

$$NR[pos] = \max\left(\frac{\sum(V_1[pos]) - V_1[pos][R[pos]]}{\sum(V_1[pos])}, \frac{\sum(V_2[pos]) - V_2[pos][R[pos]]}{\sum(V_2[pos])}\right) \tag{3.2}$$

Finally, we derive a genotype for the variant from the prediction vector V_1 and V_2. If a variant has a likelihood above a set threshold observed in both V_1 and V_2, we set the genotype as a homozygous alternate (1/1). If the likelihood is above the threshold in one vector but not in the other, we call it a heterozygous variant (0/1). For each variant, we use NR value of that position to be the genotyping quality.

PEPPER-SNP: Code availability

PEPPER-SNP is available at `https://github.com/kishwarshafin/pepper`.

PEPPER-HP: Haplotype-aware sequence prediction.

PEPPER-HP is a haplotype-aware sequence prediction tool designed to find candidate variants from read alignments. In PEPPER-HP, we take a phased alignment file as input where each read has a haplotag of $\{0, 1, 2\}$ indicating which haplotype the read represents (or lack of haplotype information), and we output a set of SNP, insertion, and deletion candidates for genotyping using DeepVariant.

Similar to the PEPPER-SNP submodule, PEPPER-HP has three steps, image generation, inference, and candidate finding. In the image generation step, we generate two sets of summary statistics, one per haplotype, and save them as image-like tensors. We use a recurrent neural network to predict bases on each haplotype for each genomic location in the inference step. Finally, we calculate likelihood values for SNP and INDEL candidates based on base-predictions of each haplotype. We consider candidates with likelihood values over a certain threshold to be candidate variants and propose them to DeepVariant for genotyping. Similarly, likelihood thresholds for PEPPER-HP were determined by training the model on HG002 chr1-19, selecting an appropriate threshold, and verifying on chr20.

PEPPER-HP: Image generation

In the image generation step of PEPPER-HP, the input is an alignment file with phased reads, and we generate image-like summary statistics of base-level information per genomic location for each haplotype $\{0,1\}$. The summary of haplotype 1 provides weighted observation of bases from reads with haplotag 1 divided into nucleotide and orientation. Similarly, the haplotype 2 summary provides weighted observations of bases from reads with haplotag 2. Reads that are unphased or have haplotag 0 contribute to summary statistics for both haplotypes.

In PEPPER-HP, we represent position in reference sequence using two values: position and index. The position value indicates a location in the reference sequence, and we use the index to accommodate insert alleles anchored to a position. All reference sequence positions have an index of 0. On each haplotype $\{0,1\}$, we iterate over all haplotype associated reads that align a genomic location and encode ten features to encode base-level information: $\{A, C, G, T, Gap(*)\}$ divided into two read orientations: $\{forward, reverse\}$. The weights depend on the mapping quality and base quality of the associated reads. Finally, we normalize the weights of each genomic position based on the haplotype associated read coverage.

PEPPER-HP: Inference model

In Supplemental figure A.9, we describe the feature encoding scheme of PEPPER-HP. We derive two summary statistics based on the haplotype association of the reads. The top row of the image, annotated as REF, describes the reference base observed at each genomic location. The colors describing the bases are $\{A : Blue, C : Red, G : Green, T : Yellow, Gap(*) : White\}$. Each column represents a reference position with two values $(pos, index)$. For example, pos $(14, 0)$ is the reference reference position 14 and $(14, 1)$ is the insert base anchored in position $(14, 0)$. For each haplotype, we use ten features to encode base-level information: $\{A, C, G, T, Gap(*)\}$ divided into two read orientations: $\{forward, reverse\}$. From the summary, we can see that at location $(23, 0)$ HP-1 observers C bases where the reference is T but in HP-2 the reads observe C bases that match with the reference, denoting a heterozygous variant present in haplotype-1 sequence.

We use a recurrent neural network for sequence prediction on haplotype-specific images of PEPPER-HP. The network architecture consists of two bidirectional gated recurrent unit (GRU) layers and a linear transformation layer. For each haplotype, the linear transformation layer predicts a base for each genomic location present in the image-like tensor. We use five class-labels for base prediction: $\{A, C, G, T, Gap(*)\}$.

The haplotype-specific images in 1kb chunks, and we use a sliding window method to go over the 1kb chunk in overlapping 100bp segments on both haplotype images simultaneously. We start from the leftmost window of 100bp and slide the window to 50bp to the right for the next step. We use two global counters from the base predictions, one per haplotype, to record the haplotype-specific base predictions.

The inference scheme of PEPPER-HP is shown in Supplemental figure A.10. We have two haplotype-specific image for each genomic region representing two haplotypes. For each haplotype, we start from the leftmost window and generate haplotype-specific base predictions. The base predictions are recorded in two global counters.

We train the PEPPER-HP inference model using a gradient descent method. We use adaptive moment estimation (Adam) to compute gradients based on a cross-entropy loss function. The loss function is defined to calculate the prediction performance at each genomic location against a labeled set of expected base observations derived from the Genome-In-A-Bottle (GIAB) truth set. We use GIAB v3.3.2 truth set as the variants in v3.3.2 are phased. We use phase-specific base predictions to optimize PEPPER-HP model for each haplotype.

We train PEPPER-HP with three sets of HG002 data with 50×, 80×, and 100× coverage. We further generate multiple train sets by arbitrarily downsampling the three training sets at different fractions. We split the training sets into three sets: train, test, and holdout. We use chromosome 1 to 19 for training, chromosome 20 for testing, and we keep 21 and 22 as holdout sets. We train the models for several epochs and test after each epoch. Finally, we pick the model that performs the best on the holdout dataset.

PEPPER-HP: Candidate finding

In the candidate finding step of PEPPER-HP, we evaluate variants observed from the read alignments. We use the base-predictions from the neural network to calculate the likelihood of an observed allele. If the likelihood value of a candidate variant is above a set threshold, we pick that variant as a potential candidate for DeepVariant to assess.

We evaluate the SNPs and INDELs observed in read alignments to find potential candidate variants. First, at each genomic location $(pos, index)$ we use the haplotype-specific prediction values of base-classes $V_{H1}[pos, index] = [A_1, C_1, G_1, T_1, Gap(*)_1]$ and $V_{H2}[pos, index] = [A_2, C_2, G_2, T_2, Gap(*)_2]$. We iterate over all the reads to find potential variants by recording the read base that does not match the reference base. We calculate the allele likelihood of a SNP candidate AP_{SNP} with base observation b, observed at position $(pos, index)$, by taking the maximum likelihood from the prediction vectors $V_{H1}[(pos, index)]$ and $V_{H2}[(pos, index)]$ as described in equation 3.3. For inserts and deletes we extend the likelihood calculation to cover the length of the allele.

$$AP_{SNP}[b, pos] = max(V_{H1}[(pos, 0)][b], V_{H2}[(pos, 0)][b]) \tag{3.3}$$

We also calculate the likelihood of observing a variant other than the reference allele at any location. In equation 3.4, let $R[(pos, index)]$ be the reference base at location $(pos, index)$ and N be the maximum index value observed in position pos. The reference base at any position with $index > 0$ is $gap(*)$. We take the maximum value of observing a non-reference base between $(pos, 0)$ and $(pos, max\_index[pos])$. For each index, we calculate the total value of the prediction vector, subtract the observation likelihood of the reference base and divide by the sum of the prediction vector. The non-reference observation likelihood NR is the maximum value we observe across index values. For insertion and deletion alleles we cover the allele length and take the maximum value as the NR for those candidates.

$$NR[pos] = \max_{i=0}^{N}\left(\frac{\sum(V_{H1}[(pos,i)]) - V_{H1}[(pos,i)][R[pos,i]]}{\sum(V_{H1}[(pos,i)])}, \frac{\sum(V_{H2}[(pos,i)]) - V_{H2}[(pos,i)][R[pos,i]]}{\sum(V_{H2}[(pos,i)])}\right) \tag{3.4}$$

Based on the allele likelihood and non-reference observation likelihood, we calculate a likelihood value for each type of candidate SNP, insert and delete. Then for each type, we set a threshold value and if a candidate passes the threshold value, we propose the candidate to DeepVariant for genotyping.

PEPPER-HP: Code availability

PEPPER-HP is available at `https://github.com/kishwarshafin/pepper` as a sub-module.

Margin

Margin is a suite of tools employing Hidden Markov Models (HMMs) to perform genomic analysis with long reads. MarginPhase (the first module) was introduced alongside

WhatsHap as a tool performing joint genotyping and phasing [46]. MarginPolish (the second module) was introduced as a graph-based assembly polisher which can do standalone polishing and is the first step in a two-part polishing framework MarginPolish-HELEN [194]. Release 2.0 of Margin incorporates both tools into one suite, including diploid-aware polishing in MarginPolish which has informed improvements in a new iteration of MarginPhase.

In this paper we focus exclusively on improvements made to the phasing submodule. The core partitioning algorithm is described in our previous work [46], but we provide a summary here of the previous methodology, followed by a description of the modifications made in the current iteration. First, we give a high-level overview of the phasing workflow.

Overview

Margin takes as input an alignment (BAM), reference (FASTA), and variant set (VCF). It determines regions to run on from the alignment and variant set, and it breaks the input into chunks to enable multiprocessing. For each chunk, it extracts reference substrings around each variant site, and read substrings aligning around each variant site. For each variant site, it calculates the alignment likelihood between all substrings and all variant alleles. These likelihoods are used in the core phasing algorithm which bipartitions reads and assigns alleles to haplotypes. After all chunks have been analyzed, we stitch the chunks together to produce results across whole contigs. Last, we output a copy of the input BAM with haplotagged reads and a copy of the input VCF with phased variants.

Parameterization. Margin is parameterized with a configuration file. For each configuration, parameters are grouped into `polish` and `phase` sections. There is overlap in the parameters used by both submodules, so some parameters used by the phasing algorithm fall under the `polish` heading and vice versa. When referenced in this document, we specify the full path of the parameter and the default value associated with the intended configuration. Default parameter values and thresholds were determined by experimentation on the HG002 sample.

Core Phasing Algorithm

For the core phasing algorithm, we construct a graph $G = (V_G, E_G)$ describing all possible bipartitions of reads with positions as variant sites, vertices as a combination of position and read bipartitions, and edges as possible transitions between bipartitions for adjacent positions. For example, at position P_i with aligned reads R_1 and R_2, we have the possible vertices V_{i_0} with haplotypes $H_1 = \{R_1, R_2\}$ and $H_2 = \{\}$ $(R_1, R_2/.)$, $V_{i_1} = R_1/R_2$, $V_{i_2} = R_2/R_1$, and $V_{i_3} = ./R_1, R_2$. At position P_j with the same aligned reads R_1, R_2 and a new read R_3, vertex $V_{j_0} = R_1, R_2, R_3/.$ and $V_{j_1} = R_1, R_2/R_3$ are both connected to vertex V_{i_0} because all reads shared between vertices are have the same haplotype assignment, but $V_{j_2} = R_1/R_2, R_3$ is not connected to V_{i_0} because read R_2 has different haplotype assignments in the two vertices. We extend each vertex as described above to additionally represent all possible genotypes. After running the forward-backward algorithm on this graph, at each position the posterior distribution over states describing read bipartitions and genotypes can be marginalized to determine the most likely genotype.

The state space for this algorithm increases exponentially with the number of reads at each position. To account for this, Margin implements a pruning and merging heuristic where the input is divided into smaller pieces, unlikely states are pruned, and the resulting graphs are merged before running the full forward-backward algorithm.

Improved Functionality

One of the most significant changes to the Margin workflow is that we now only analyze sites proposed by the input VCF. Previously we considered any reference position where less than 80% of the nucleotides agreed with the reference base as a candidate variant site. To determine which proposed variants are considered, we read the input VCF and remove all INDEL variants (`phase.onlyUseSNPVCFEntries = false`), all homozygous variants (`phase.includeHomozygousVCFEntries = false`), all non-PASS variants (`phase.onlyUsePassVCFEntries = true`), and all variants with a quality score below `phase.minVariantQuality = 10`. For each chunk, we perform an adaptive sampling of variants (`phase.useVariantSelectionAdaptiveSampling = true`) where we start by taking all variants with a quality score above `phase.variantSelectionAdaptiveSamplingPrimaryThreshold = 20`. If the average distance between these variants for the chunk is greater than `phase.variantSelectionAdaptiveSamplingDesiredBasepairsPerVariant = 2000`, we take variants ordered by quality score descending until we achieve the desired number of variants. These values were determined after experimentation on HG002.

Instead of considering only the nucleotide aligned directly to the variant position as the first iteration of Margin had done, we now extract substrings from the reference and reads and perform an alignment to determine which allele the read most likely originated from. In theory, this allows Margin to use INDELs during phasing, although for our current evaluations we do not test this functionality. When extracting reference bases, we take `phase.columnAnchorTrim = 12` bp from the reference before the variant position, and the same amount upstream from the end of the variant position. All alleles (including the reference allele) in the VCF are recreated by substituting the replaced reference base with the allelic sequence from the VCF. Note that there is no further modification of the reference sequences for cases where multiple variants fall within the extracted region. From the reads, we extract all sequence between the first and last position matched to the reference over the extracted reference region. Reads substrings for which the average base-level quality score over the substring is less than `polish.minAvgBaseQuality = 10` are excluded at these positions.

The original Margin phasing HMM used an emission model based off of a read error model (empirically determined) and a mutation likelihood model (based on likelihood of specific mutations). In the new implementation, we replace this with a likelihood generated by aligning the read substrings to the allelic substrings.

As in the previous iteration, after running the forward-backward algorithm on the HMM we marginalize over possible genotypes at each variant site to determine final predicted haplotypes. Given these haplotypes and the read-to-allele alignment likelihoods, we compute from which haplotype the read most likely originated using the joint probability of the read substrings aligning to the alleles for each haplotype. This haplotyping step is performed for all reads used for phasing, as well as other reads filtered out before phasing (as described below in the Chunking section). If a read does not span any variants or has equal likelihood of aligning to both haplotypes, no prediction is made regarding the haplotype from which it originated.

Chunking

Margin divides input into `polish.chunkSize = 100000` bp chunks with `polish.chunkBoundary = 10000` bp boundary on both ends, resulting in 2x `polish.chunkBoundary` overlap between chunks. Once the boundaries have been determined, the ordering of the chunks is mutated (`polish.shuffleChunks = true`) by default based on descending order of size (`polish.shuffleChunksMethod = size_desc`), with random ordering (`random`) also as a configurable option. While the ordering does not have a large effect on the runtime during phasing, we found that deep chunks would take drastically longer to complete for polishing, and ensuring that they were completed first would reduce overall runtime for the submodule.

When operating on a chunk, Margin first extracts all the reads that have an alignment between the start and end of the chunk, tracking all alignments falling between the start and end of the extended boundary. Margin then collects a set of reads to run the main algorithm on, first by removing reads that have a MQ score below `polish.filterAlignmentsWithMapQBelowThisThreshold = 10`, are secondary alignments (`polish.includeSecondaryAlignments = false`), or are supplementary alignments (`polish.includeSupplementaryAlignments = false`). Given this set of reads, we downsample to an expected depth of `polish.maxDepth = 64` for haplotagging and `polish.maxDepth = 32` for variant phasing. The downsampling is biased to maximize coverage over heterozygous variants given the constraint on expected coverage. To accomplish this, we compute the sampling probabilities for each read according to the following linear program, which we solve using the LP Solve library (`http://lpsolve.sourceforge.net/`).

$$\begin{aligned} \max_{p} \quad & \sum_{i=1}^{N} v_i p_i \\ \text{subject to} \quad & 0 \leq p \leq 1 \\ & \sum_{i=1}^{N} l_i p_i \leq CL, \end{aligned} \tag{3.5}$$

where p_i is the probability of selecting read i, v_i is the number of heterozygous variants on read i, l_i is the length of read i, C is the desired coverage, and L is the length of the chunk.

The algorithm runs on the chunk region with the primary set of reads (all reads kept after filtering and downsampling) and assigns reads and variants to haplotypes. Then it takes any removed reads (either through filtering or downsampling) and assigns them to haplotypes as described above. Margin tracks the assignment of variants to haplotypes within the chunk (not including the boundaries), and of reads to haplotypes for the whole region (including the boundaries).

To stitch two chunks together, we need to determine whether the two previous haplotypes (P_1, P_2) are oriented with the two current haplotypes (C_1, C_2) in *cis* (P_1C_1, P_2C_2) or in *trans* (P_1C_2, P_2C_1). To do this, we compare the number of reads that are in *cis* and in *trans* between the two chunks; if there are more reads in *trans*, we switch the haplotypes of the current chunk's reads and variants. To mulithread this process, Margin separates all the chunks in a contig into `numThreads` contiguous groups. The chunks in each of the groups are stitched together by a single thread, and then the same stitching process is used to stitch each of these groups together to complete the whole contig.

The final assignment of a read to a haplotype is determined by the haplotype it was assigned to in the first chunk for the contig. The final assignment of an allele to a haplotype is determined by the chunk it falls within (boundary region excluded).

Phaseset Determination

When writing the output VCF, Margin makes predictions about which sets of variants are confidently inherited together and annotates the output with phaseset

(PS) tags. Margin will assign a phaseset to a variant if it is heterozygous, a SNP (`phase.onlyUseSNPVCFEntries = true`), and if it agrees with the genotype from the input VCF (`phase.updateAllOutputVCFFormatFields = false`). For variants meeting this criteria, as Margin iterates through the VCF it will extend the current phaseset unless *(a)* the variant is the first in the contig, *(b)* there are no reads spanning between the current variant and the previous variant, *(c)* there is an unlikely division of reads for the variant (explained below), or *(d)* the reads spanning the current variant and the previous variant are discordant above some threshold (explained below). The values described below were determined after experimentation on HG002.

To identify unlikely divisions of reads (which we take as potential evidence there is an error in the phasing), we take the number of primary reads assigned to each haplotype and find the binomial p-value for that division of reads. If that probability is less than the threshold `phase.phasesetMinBinomialReadSplitLikelihood = 0.0000001`, we create a new phase set for this variant.

Within each chunk region (boundary region excluded) and after determining haplotype assignment for the reads, we track which primary reads were used for phasing and to which haplotype they were assigned in the chunk. This serves as a check against poorly-phased or poorly-stitched chunks. To determine discordancy in the phasing between variants, we compare the number of reads which are in *cis* or "concordant" (C_c) given the read assignment to adjacent variants, and the number in *trans* or "discordant" (C_d) between variants. If the discordancy ratio $C_d/(C_c+C_d)$ is greater than `phase.phasesetMaxDiscordantRatio = 0.5`, we create a new phase set for this variant.

Margin: Code availability

Margin is available at `https://github.com/UCSC-nanopore-cgl/margin`.

Local Phasing Correctness

The local phasing correctness (LPC) is a novel metric for measuring phasing accuracy that we developed for this study. More precisely, the LPC is a family of metrics parameterized by a varying parameter $\rho \in [0, 1]$, which controls the degree of locality. The LPC can be seen as a generalization of the two most common metrics used to evaluate phasing accuracy: the switch error rate and the Hamming rate. The switch error rate corresponds to the LPC with $\rho = 0$ (fully local), and the Hamming rate is closely related to the LPC with $\rho = 1$ (fully global). With intermediate values of ρ, the LPC can measure meso-level phasing accuracy that the two existing metrics cannot quantify.

The LPC consists of a sum over all pairs of heterozygous variants where each pair contributes an amount that decays with greater genomic distance. If the variants are incorrectly phased relative to each other, the pair contributes 0. This sum is normalized by its maximum value so that the LPC is always takes a value between 0 and 1. In mathematical notation,

$$LPC_\rho = \frac{\sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} \rho^{d(i,j)} \delta(i,j)}{\sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} \rho^{d(i,j)}}, \tag{3.6}$$

where $d(i,j)$ is the distance between variants i and j, and $\delta(i,j)$ is an indicator for whether variants i and j are correctly phased relative to each other. In the case that $\rho = 0$ the above formula is undefined, and we instead take the limit of LPC_ρ as $\rho \to 0$.

If we take $d(i,j)$ to be $|i - j|$, then it can be shown that LPC_0 is equivalent to the complement of the switch error rate (i.e. the "switch correctness" rate). LPC_1 is not equivalent to the Hamming rate, but they are monotonic functions of each other. Thus, LPC_1 and Hamming rate always produce the same relative ranking but not the same numerical value. Alternatively, we can take $d(i,j)$ to be the genomic distance between the variants, measured in base pairs. In doing so, the LPC no longer has the provable relationships to switch error rate and Hamming rate, neither of which has any mechanism to incorporate genomic distance. However, we consider the genomic distance to be a more relevant measurement of distance for phasing accuracy than the variants' ordinal numbers. Accordingly, we believe that this amounts to a further strength of the LPC over existing metrics, and all LPC values reported in this work use this definition of distance.

The ρ parameter is mathematically convenient but difficult to interpret. To improve interpretability, we can reparameterize the LPC with $\lambda = -\log 2 / \log \rho$, which we call the "length scale". This is the distance (measured either in base pairs or number of variants) at which a pair of variants has 1/2 of the maximum weight. The length scale gives an approximate sense of the scale of distances that the LPC incorporates, although it is worth stressing that pairs that are closer together always receive more weight than pairs that are further apart.

Local Phasing Correctness: Code availability

The code used to calculate LPC is available at `https://github.com/UCSC-nanopore-cgl/margin`.

DeepVariant

DeepVariant is a small variant identification method based on deep neural networks that achieve high performance on different short-read and long-read sequencing platforms [166]. DeepVariant has released models for different datatypes and a details of the methods implemented in DeepVariant can be found in associated releases [166, 221]. Here we present an overview of the methods we implemented to achieve high performance with nanopore data.

Adapting DeepVariant to Oxford Nanopore reads

DeepVariant performs variant calling in three stages, “make examples”, “call variants”, and “postprocess variants”. In “make examples”, potential variant positions are identified by applying a minimal threshold for evidence. In positions meeting the candidate generation criteria, the reads overlapping the position are converted into a pileup of a

221-bp window centered at the variant. Multiple features of the reads are represented as different dimensions in the pileup, including the read base, base quality, mapping quality, strand, whether the read supports the variant, and whether the base matches the reference. Prior to this work, the heuristics for candidate generation were simple (at least 2 reads supporting a variant allele and a variant allele fraction at least 0.12). However, the higher error rate of Oxford Nanopore data generated far too many candidates for DeepVariant to call variants in a genome in a reasonable time. To integrate DeepVariant with PEPPER, we developed the ability to import candidates directly from a VCF to replace the logic in "make examples". This allows DeepVariant to read the output of PEPPER, and in theory, can be used in a similar manner with the outputs of other methods.

DeepVariant has released models for different datatypes [166, 221]. In order to achieve high performance on Oxford Nanopore data, a new DeepVariant model trained for this datatype was required. For this, we modified the training process for DeepVariant. The training process for DeepVariant is very similar to the variant calling process (generating candidates from make examples in a similar manner), but with the addition of a step which reconciles a candidate variant with the truth label from Genome-in-a-Bottle. We adapted this process to use the `VcfCandidateImporter` to propose PEPPER candidates. Because the representations of variants can diverge from the representation in GIAB (even for the same event), modifying the process for training required multiple rounds of iteration to identify mislabeled edge cases and to modify the proposed candidate representation for training.

With these modifications, training of DeepVariant models using the existing machinery could proceed. These modifications also allowed existing logic for the stages "call variants" and "postprocess variants" to work directly with the trained Oxford Nanopore model.

DeepVariant: Code availability

All code in the DeepVariant repository incorporates the improvements made for this paper (https://github.com/google/deepvariant). This repository also contains a model retraining tutorial (https://github.com/google/deepvariant/blob/r1.1/docs/deepvariant-training-case-study.md).

Training PEPPER-DeepVariant for ONT basecallers

It is necessary to train new models once the underlying read quality of Oxford Nanopore changes (i.e. basecaller update). The amount of compute resources depends on the amount of data for training, and the degree of difference between the old data for the model and the new data. It is possible to take the existing PEPPER model and re-train it with 100x HG002 on one gpu for few hours to get a model for the newer chemistry. Similarly, it is possible to take the existing DeepVariant model and train the model for few hours on one GPU to get a model for the new basecaller. A case study of re-training an Illumina model for BGISEQ is available here (https://github.com/google/deepvariant/blob/r1.1/docs/deepvariant-training-case-study.md). As the basecaller updates of ONT are incremental, we believe retraining to newer basecall sets will not have resource bottleneck.

Assembly polishing with PEPPER-Margin-DeepVariant

The assembly polishing method of PEPPER-Margin-DeepVariant is described below:

1. **PEPPER-SNP**: PEPPER-SNP finds single nucleotide polymorphisms (SNPs) from the read alignments to a haploid assembly using a RNN. For assembly polishing, we use the same infrastructure described in the haplotype-aware variant calling section.

2. **Margin**: Margin takes the SNPs reported by PEPPER-SNP and generates a haplotagged alignment file using a HMM. For assembly polishing, we use the same infrastructure described in the haplotype-aware variant calling section.

3. **PEPPER-HP**: PEPPER-HP takes the haplotagged alignment file and evaluates each haplotype independently and produces haplotype-specific candidate SNP and INDEL-like errors present in the assembly.

    - *Haplotype-1 and Haplotype-2 candidate finding*: For haplotype-1, we take reads with `HP-1` and `HP-0` and generate base-level summary statistics. Then we use a RNN to produce nucleotide base predictions for each location of the genome. Finally, we find all the observed SNP or INDEL-like candidates in the reads with `HP-1` and `HP-0` tag and calculate a likelihood of each candidate to be a potential error in the assembly. If the likelihood of the candidate is above a set threshold, we propose that candidate as a potential edit for haplotype-1(`HP-1`) of the assembly. For haplotype-2, we take reads with `HP-2` and `HP-0` and find SNP and INDEL-like candidate errors following the same process described in Haplotype-1 candidate finding.

4. **DeepVariant**: DeepVariant takes the haplotype-specific candidate set from PEPPER-HP and the haplotagged alignment from Margin to identify errors present in the assembly using a convolutional neural network (CNN).

   - *Haplotype-1 and haplotype-2 polishing*: For each candidate error from haplotype-1 set proposed by PEPPER-HP, we generate a feature set representing base, base-quality, mapping quality etc. in a tensor like format. In haplotype-specific polishing setup, reads are sorted by their haplotags in the order `{HP-1, HP-0, HP-2}`. Then we use a pre-trained `inception_v3` CNN to generate a prediction of `{0/0, 1/1}` for each candidate. Candidates in haplotype-1 that are classified as `{1/1}` are considered as missing heterozygous variants from the haploid assembly or an error present in the assembly. We apply the candidates of haplotype-1 classified as `{1/1}` to the haploid assembly using `bcftools consensus` to get a polished assembly `haplotype_1.fasta` representing one haplotype of the sample.

For haplotype-2, we sort the reads by their haplotags in the order `{HP-2, HP-0, HP-1}`. Candidates in haplotype-2 that are classified as `{1/1}` are similarly considered to be a missing heterozygous variant or an error present in the assembly, and are applied to the haploid assembly to get the second polished haplotype (`haplotype_2.fasta`).

We trained the PEPPER-Margin-DeepVariant assembly polishing method with HG002 assembly generated by the Shasta assembler. We first aligned the HG002 Shasta assembly to GRCh38 reference with dipcall[112] to associate assembly contigs with GRCh38 reference. Then we aligned HG002 GIAB v4.2.1 small variant benchmarking set to the assembly and marked the high-confidence regions to restrict training only in regions that GIAB notes as high-quality. The alignment between HG002 GIAB v4.2.1 to the assembly produced the training set. We trained PEPPER and DeepVariant on contigs that associate to `chr1-chr19` and used `chr20` as a holdout set. We used the same approach for Oxford Nanopore and PacBio HiFi based models.

Part IV

Efficient de novo assembly of eleven human genomes in nine days

Chapter 4

Nanopore sequencing and Shasta toolkit enables de novo assembly of eleven human genomes in nine days.

Preamble

This chapter contains the text from "Nanopore sequencing, and the Shasta toolkit enables efficient de novo assembly of eleven human genomes," published in Nature Biotechnology in May 2020. The manuscript details a *de novo* assembly and polishing pipeline aimed at a large cohort of samples. The core contribution to this manuscript comes from Paolo Carnevali, who developed the Shasta assembler, which holds the core results of this manuscript. Paolo Carnevali shares the corresponding authorship of this manuscript with Miten Jain and Benedict Paten.

I share the first authorship of this manuscript with Trevor Pesout, Ryan Lorig-Roach, Marina Haukness, and Hugh E. Olsen. My contribution in this manuscript is designing the polisher HELEN and performing analysis between the assemblies. This manuscript received contributions from Colleen Bosworth, Joel Armstrong, Kristof Tigyi, Nicholas Maurer, Sergey Koren, Fritz J. Sedlazeck, Tobias Marschall, Simon Mayes, Vania Costa, Justin M. Zook, Kelvin J. Liu, Duncan Kilburn, Melanie Sorensen, Katy M. Munson, Mitchell R. Vollger, Jean Monlong, Erik Garrison, Evan E. Eichler, Sofie Salama, David Haussler, Richard E. Green, Mark Akeson, Adam Phillippy, Karen Miga. This manuscript was the first demonstration of the multiplexed sequencing ability of the nanopore promethION device. I consider this a significant achievement to showcase the ability to scale nanopore sequencing to cohort level.

Introduction

Short-read sequencing reference-assembly mapping methods only assay about 90% of the current reference human genome assembly [46], and closer to 80% at high-confidence [236]. 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 [166]. However, short reads are much less able to *de novo* assemble a new genome [14], to discover structural variations (SVs) [3, 97] (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 [24].

Third generation sequencing technologies, including linked-reads [10, 53, 220] and long-read technologies [87, 50], 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 [46, 82, 187, 163], long-read technologies can generate highly contiguous *de novo* genome assemblies [29].

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 [88]. 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 [49, 56, 90]. We contributed to the recent consortium-wide effort to perform the *de novo* assembly of a nanopore sequencing based human genome [88]. 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 [10]. 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.

Results

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) [34] and Genome-in-a-Bottle (GIAB) [232] 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 <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).

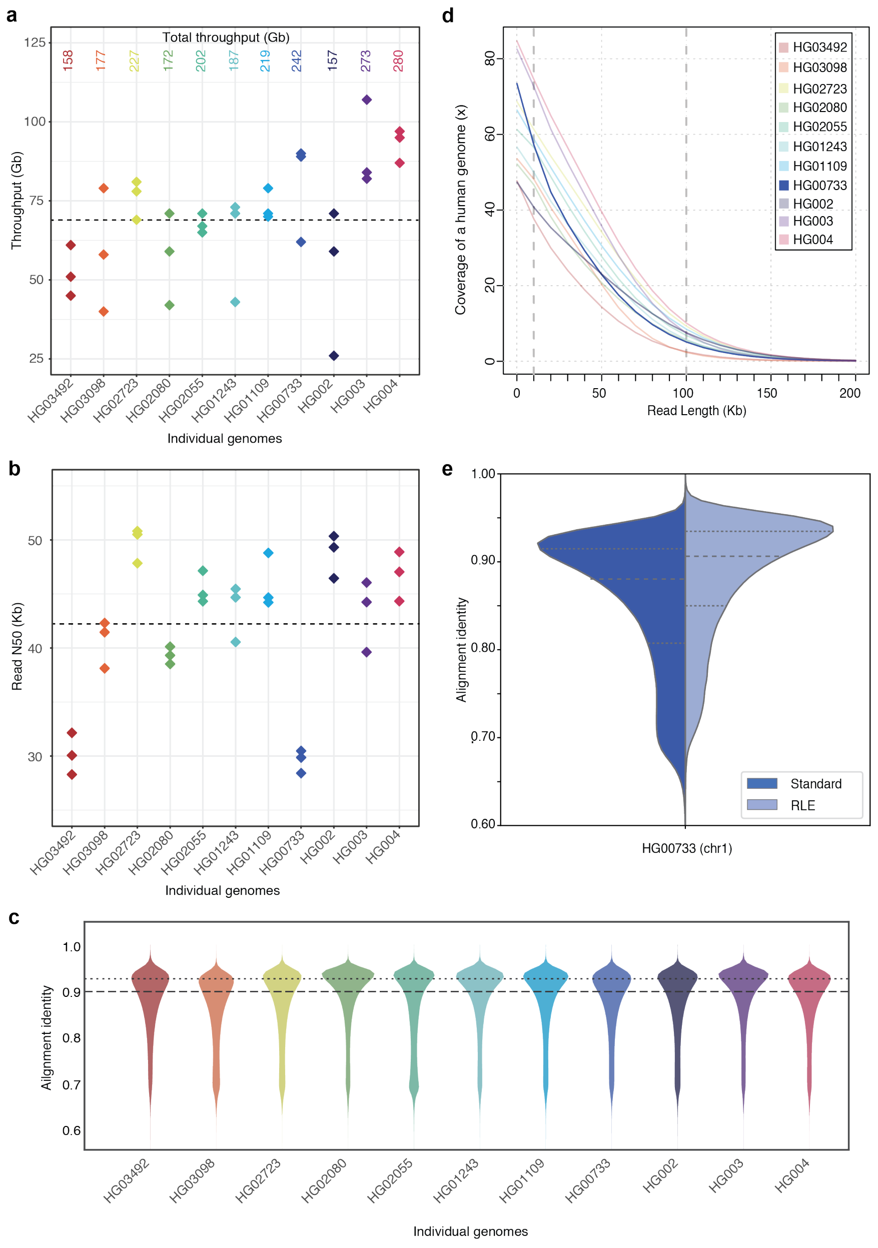


Figure 4.1: **Nanopore sequencing results. (a)** Throughput in gigabases from each of three flowcells for eleven samples, with total throughput at top. **(b)** Read N50s for each flowcell. **(c)** Alignment identities against GRCh38. Medians in a, b and c shown by dashed lines, dotted line in c is mode. **(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.

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 B.1, B.2, and B.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 B.4), extrapolating both are within the lower range achieved with cell line derived DNA.

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 [88]. 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) [110, 181, 140]. 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 [87]. 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* [17, 11] 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.

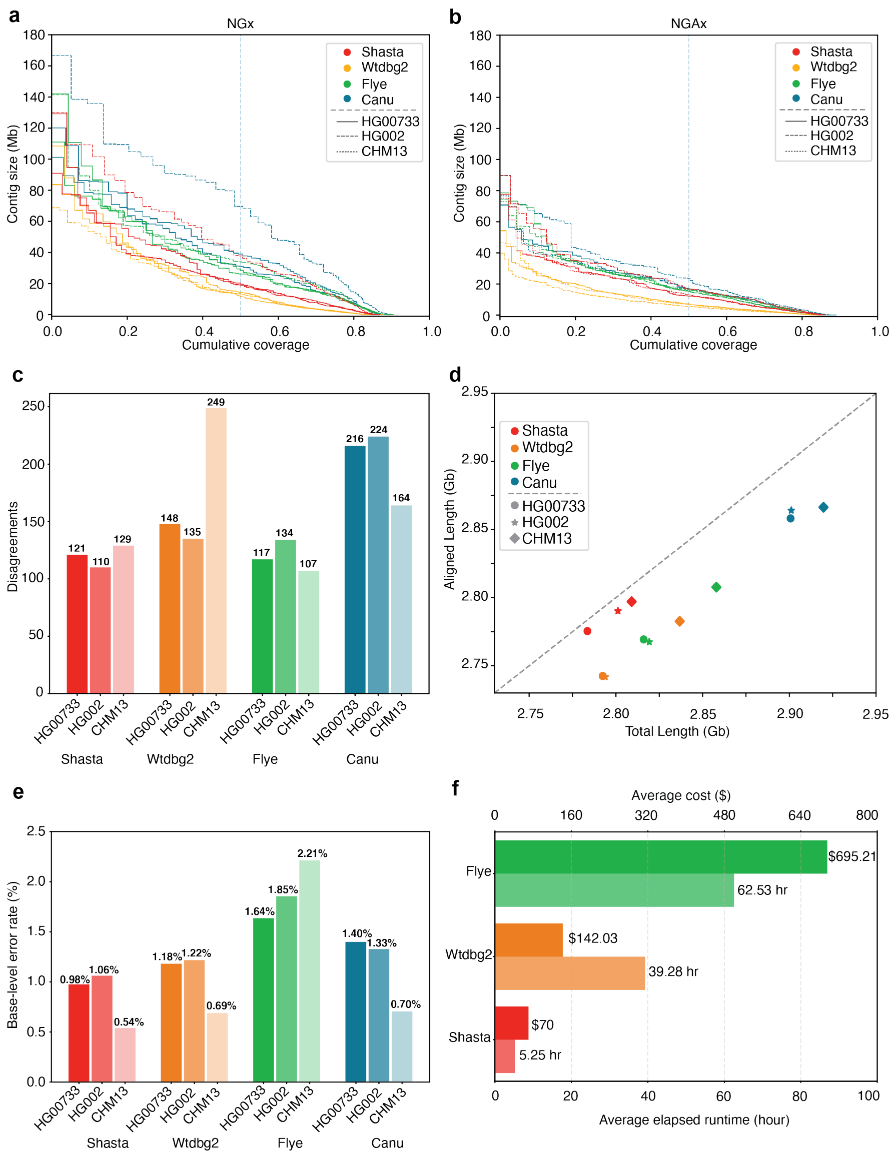


Figure 4.2: **Assembly results for four assemblers and three human samples, before polishing. (a)** NGx plot showing contig length distribution. The intersection of each line with the dashed line is the NG50 for that assembly. **(b)** NGAx plot showing the distribution of *aligned* contig lengths. Each 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).

To validate Shasta, we compared it against three contemporary assemblers: Wtdbg2 [182], Flye [93] and Canu [96]. 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 [206].

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.

Conversely, aligning the samples to GRCh38 and evaluating with QUAST [138], 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 B.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 B.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[6, 202]); in the case of HG002, we further excluded a set of known

SVs [234]. 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 B.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. B.7). Finally, we used QUAST to calculate disagreements between the T2T Consortium's chromosome X assembly, a highly curated, validated assembly [206] 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 B.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 B.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 B.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 B.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 B.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 B.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 B.14, B.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. B.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 B.16).

Contiguously assembling MHC haplotypes

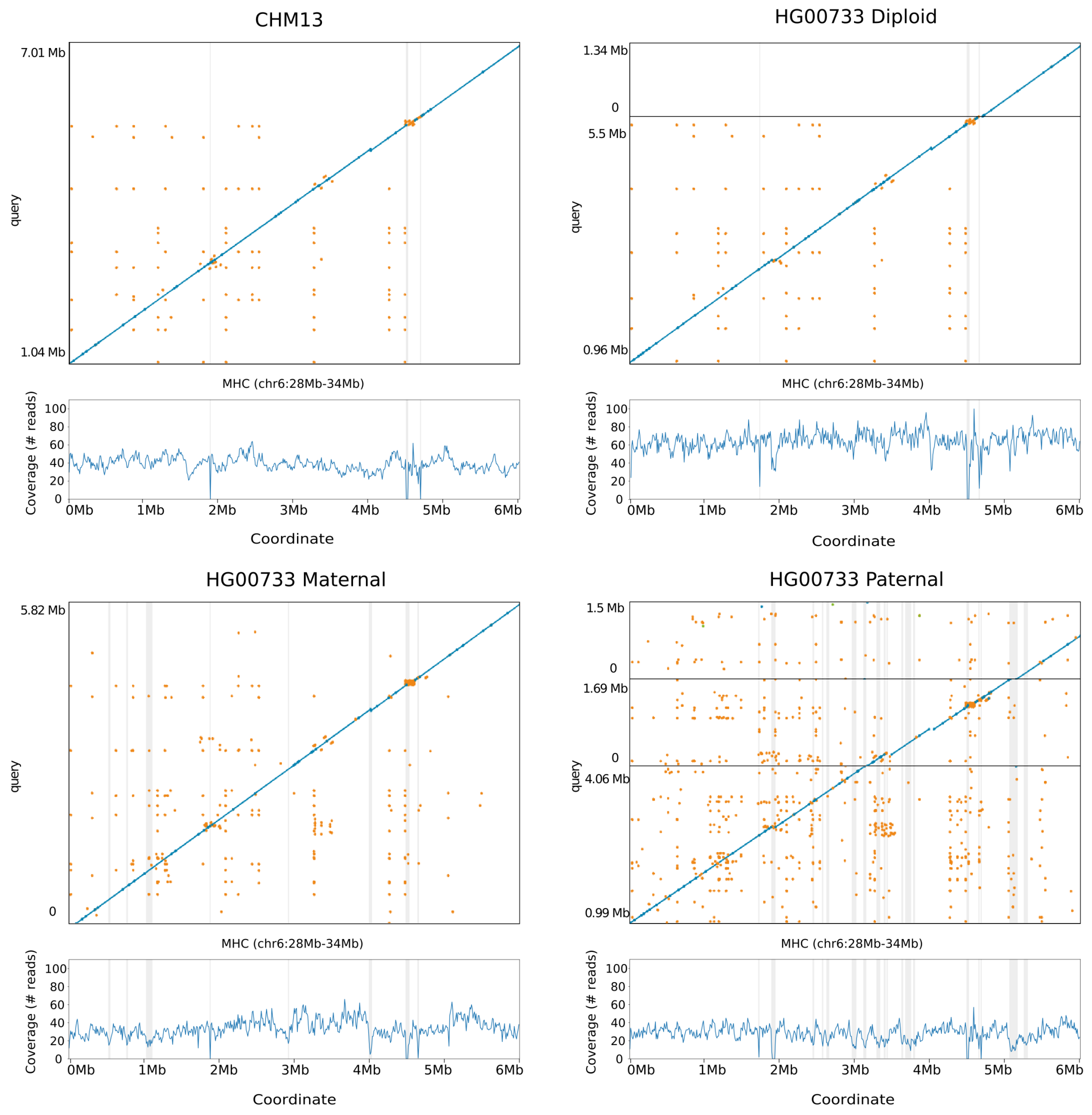


Figure 4.3: **Shasta MHC assemblies vs GRCh38.** Unpolished Shasta assembly for CHM13 and HG00733, including HG00733 trio-binned maternal and paternal assemblies. Shaded gray areas are regions in which coverage (as aligned to GRCh38) drops below 20. Horizontal black lines indicate contig breaks. Blue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

The Major Histocompatibility Complex (MHC) region is difficult to resolve using short reads due to its repetitive and highly polymorphic nature [15], but recent efforts to apply long read sequencing to this problem have shown promise [88, 207]. 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. B.2 and Supplementary Table B.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. B.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 [95], 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 B.18). When using haploid trio-binned assemblies, the MHC was spanned by a single contig for the maternal haplotype (Fig. 4.3c, Supplementary Fig. B.4, Supplementary Table B.19), 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.

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 [44]. From these statistics MarginPolish generates a weighted RLE Partial Order Alignment (POA) graph [104] 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) [131] 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 [210] followed by Medaka [123]. 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 B.20, B.21 and B.22 detail error rates for the four methods performed on the HG00733 and CHM13 Shasta assemblies (see Online Methods) using Pomoxis [124]. 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 [206]. 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. B.5).

Figure 4.4c and Supplementary Table B.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 [218]. 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 B.24). Notably, attempting to use Pilon polishing on the raw Shasta assembly resulted in much poorer results (QV24).

Figure 4.4d and Supplementary Table B.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 B.26).

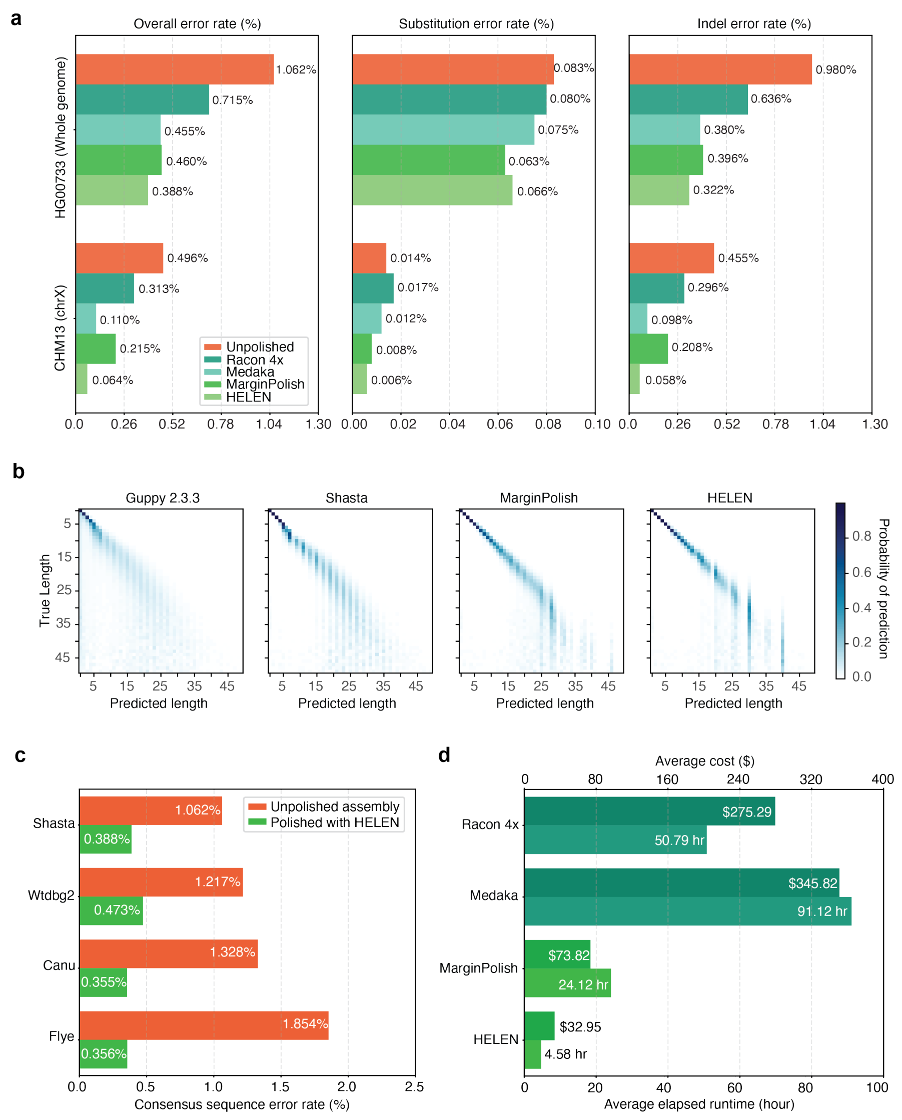


Figure 4.4: **Polishing Results.** **(a)** Balanced error rates for the four methods on HG00733 and CHM13. **(b)** Row-normalized heatmaps describing the predicted run-lengths (x-axis) given true run lengths (y-axis) for four steps of the pipeline on HG00733. **(c)** Error rates for MarginPolish and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

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 [55], which can annotate a genome assembly using the human GENCODE [59] reference human gene set (Table 4.1, Online Methods, Supplementary Tables B.27, B.28, B.29, and B.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 [197] using the eukaryote set of orthologs on each assembly, a smaller set of 303 expected single-copy

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.

genes (Supplementary Tables B.31 and B.32). We find comparable performance between the assemblies, with small differences largely recapitulating the pattern observed by the larger CAT analysis.

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 [215]; 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. B.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. B.7, Supplementary Table B.33). Predictably, the HiFi assembly has a higher QV value than the polished Shasta assembly (QV41 vs. QV32).

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

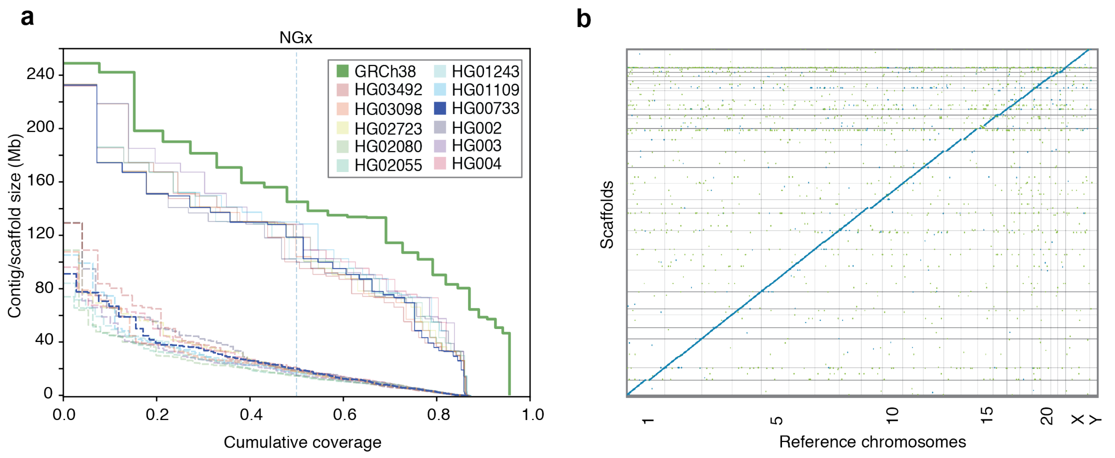


Figure 4.5: **HiRise scaffolding for 11 genomes. (a)** NGx plots for each of the 11 genomes, before (dashed) and after (solid) scaffolding with HiC sequencing reads, GRCh38 minus alternate sequences is shown for comparison. **(b)** Dot plot showing alignments between the scaffolded HG00733 Shasta assembly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

To achieve chromosome length sequences we scaffolded all of the polished Shasta assemblies with HiC proximity-ligation data using HiRise [172] (see Online Methods, Fig. 4.5a). On average, 891 joins were made per assembly. This increased the 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 B.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. B.8).

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 [88]. 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 [50, 222]. 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 [88, 206].

Relative to alternate long-read and linked-read sequencing, the read identity of nanopore reads has proven lower [87, 88]. However, original reports of 66% identity [87] 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 [222] and 10x Chromium [126], 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 [106]. This cost reduction is still clearly underway. The first MinION human assembly cost ~$40,000 in flow cells and reagents [88]. 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 [152, 108].

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 [174] 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 [29, 63, 107]. 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 [46], and the HELEN model could be improved to include haplotagged features for the identification of heterozygous sites.

Code Availability

The three novel software tools, Shasta (`https://github.com/chanzuckerberg/shasta`), MarginPolish (`https://github.com/UCSC-nanopore-cgl/marginPolish`), and HELEN (`https://github.com/kishwarshafin/helen`) are publicly available. They have open source MIT license which fully supports the open source initiative.

Online Methods

Sample selection

The goal of sample selection was to select a set of individuals that collectively captured the maximum amount of weighted allelic diversity [186]. To do this, we created a list of all low-passage lymphoblastoid cell lines that are part of a trio available from the 1000 Genomes Project collection [35] (We selected trios to allow future addition of pedigree information, and low-passage line to minimize acquired variation). In some cases, we considered the union of parental alleles in the trios due to not having genotypes for the offspring. Let a weighted allele be a variant allele and its frequency in the 1000 Genomes Project Phase 3 VCF. We selected the first sample from our list that contained the largest sum of frequencies of weighted alleles, reasoning that this sample should have the largest expected fraction of variant alleles in common with any other randomly chosen sample. We then removed the variant alleles from this first sample from the set of variant alleles in consideration and repeated the process to pick the second sample, repeating the process recursively until we had selected seven samples. This set greedily, heuristically optimizes the maximum sum of weighted allele frequencies in our chosen sample subset. We also added the three Ashkenazim Trio samples and the Puerto Rican individual (HG00733). These four samples were added for the purposes of comparison with other studies that are using them [232].

Cell culture

Lymphoblastoid cultures for each individual were obtained from the Coriell Institute Cell Repository (`coriell.org`) and were cultured in RPMI 1640 supplemented with 15% fetal bovine serum (Life Technologies). The cells underwent a total of six passages (p3+3). After expansion, cells were harvested by pelleting at 300xg for 5 minutes. Cells were resuspended in 10 ml PBS and a cell count was taken using a BiRad TC20 cell counter. Cells were aliquoted into 50 ml conical tubes containing 50 million cells, pelleted as above and washed with 10 ml PBS before a final pelleting after which the PBS was removed and the samples were flash frozen on dry ice and stored at -80°C until ready for further processing.

DNA extraction and size-selection

We extracted high-molecular weight (HMW) DNA using the QIAGEN Puregene kit. We followed the standard protocol with some modifications. Briefly, we lysed the cells by adding 3 ml of Cell Lysis Solution per 10 million cells, followed by incubation at 37°C for up to 1 hour. We performed mild shaking intermediately by hand, and avoided vortexing. Once clear, we split the lysate into 3 ml aliquots and added 1 ml of Protein Precipitation Solution to each of the tubes. This was followed by pulse vortexing three times for five seconds each time. We next spun this at 2000 x g for 10 minutes. We added the supernatant from each tube to a new tube containing 3 ml of isopropanol, followed by 50x inversion. The HMW DNA precipitated and formed a dense thread-like jelly. We used a disposable inoculation loop to extract the DNA precipitate. We then dipped the DNA precipitate, while it was on the loop, into ice-cold 70% ethanol. After this, the DNA precipitate was added to a new tube containing 50-250 μl 1x TE buffer. The tubes were heated at 50°C for 2 hours and then left at room temperature overnight to allow resuspension of the DNA. The DNA was then quantified using Qubit and NanoDrop.

We used the Circulomics Short Read Eliminator (SRE) kit to deplete short-fragments from the DNA preparation. We size-selected 10 μg of DNA using the Circulomics recommended protocol for each round of size-selection.

Nanopore sequencing

We used the SQK-LSK109 kit and its recommended protocol for making sequencing libraries. We used 1 μg of input DNA per library. We prepared libraries at a 3x scale since we performed a nuclease flush on every flow cell, followed by the addition of a fresh library.

We used the standard PromethION scripts for sequencing. At around 24 hours, we performed a nuclease flush using the ONT recommended protocol. We then re-primed the flow cell, and added a fresh library corresponding to the same sample. After the first nuclease flush, we restarted the run setting the voltage to -190 mV. We repeated the nuclease flush after another around 24 hours (i.e. around 48 hours into sequencing), re-primed the flow cell, added a fresh library, and restarted the run setting the run voltage to -200 mV.

We performed basecalling using Guppy v.2.3.5 on the PromethION tower using the GPUs. We used the MinION DNA flipflop model (`dna_r9.4.1_450bps_flipflop.cfg`), as recommended by ONT.

Chromatin Crosslinking and Extraction from Human Cell Lines

We thawed the frozen cell pellets and washed them twice with cold PBS before resuspension in the same buffer. We transferred Aliquots containing five million cells by volume from these suspensions to separate microcentrifuge tubes before chromatin crosslinking by addition of paraformaldehyde (EMS Cat. No. 15714) to a final concentration of one percent. We briefly vortexed the samples and allowed them to incubate at room temperature for fifteen minutes. We pelleted the crosslinked cells and washed them twice with cold PBS before thoroughly resuspending in lysis buffer (50 mM Tris-HCl, 50 mM NaCl, 1 mM EDTA, 1% SDS) to extract crosslinked chromatin.

The Hi-C Method

We bound the crosslinked chromatin samples to SPRI beads, washed three times with SPRI wash buffer (10 mM Tris-HCl, 50 mM NaCl, 0.05% Tween-20), and digested by DpnII (20 U, NEB Catalog No. R0543S) for 1 hour at 37°C in an agitating thermal mixer. We washed the bead-bound samples again before incorporation of Biotin-11-dCTP (ChemCyte Catalog No. CC-6002-1) by DNA Polymerase I, Klenow Fragment (10 U, NEB Catalog No. M0210L) for thirty minutes at 25°C with shaking. Following another wash, we carried out blunt-end ligation by T4 DNA Ligase (4000 U, NEB Catalog No. M0202T) with shaking overnight at 16°C. We reversed the chromatin crosslinks, digested the proteins, eluted the samples by incubation in crosslink reversal buffer (5 mM CaCl 2 , 50 mM Tris-HCl, 8% SDS) with Proteinase K (30 μg, Qiagen Catalog No. 19133) for fifteen minutes at 55°C followed by forty-five minutes at 68°C.

Sonication and Illumina Library Generation with Biotin Enrichment

After SPRI bead purification of the crosslink-reversed samples, we transferred DNA from each to Covaris® microTUBE AFA Fiber Snap-Cap tubes (Covaris Cat. No. 520045) and sonicated to an average length of 400 ± 85 bp using a Covaris® ME220 Focused-Ultrasonicator™. Temperature was held stably at 6°C and treatment lasted sixty-five seconds per sample with a peak power of fifty watts, ten percent duty factor, and two-hundred cycles per burst. The average fragment length and distribution of sheared DNA was determined by capillary electrophoresis using an Agilent® FragmentAnalyzer 5200 and HS NGS Fragment Kit (Agilent Cat. No. DNF-474-0500). We ran sheared DNA samples twice through the NEBNext® Ultra™ II DNA Library Prep Kit for Illumina® (Catalog No. E7645S) End Preparation and Adaptor Ligation steps with custom Y-adaptors to produce library preparation replicates. We purified ligation products via SPRI beads before Biotin enrichment using Dynabeads® MyOne™ Streptavidin C1 beads (ThermoFisher Catalog No. 65002). We performed indexing PCR on streptavidin beads using KAPA HiFi HotStart ReadyMix (Catalog No. KK2602) and PCR products were isolated by SPRI bead purification. We quantified the libraries by Qubit™ 4 fluorometer and FragmentAnalyzer 5200 HS NGS Fragment Kit (Agilent Cat. No. DNF-474-0500) before pooling for sequencing on an Illumina HiSeq X at Fulgent Genetics.

Analysis methods

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` [110] 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.

Base-level error-rate analysis with Pomoxis

We analyzed the base-level error-rates of the assemblies using the assess_assembly tool of Pomoxis toolkit developed by Oxford Nanopore Technology https://github.com/nanoporetech/pomoxis. We further modified the program to avoid large insertions and deletions (>50bp) and submitted a merge request https://github.com/nanoporetech/pomoxis/pull/37. The assess assembly tool is tailored to compute the error rates in a given assembly compared to a truth assembly. It reports an identity error rate, insertion error rate, deletion error rate, and an overall error rate. The identity error rate indicates the number of erroneous substitutions, the insertion error rate is the number of incorrect insertions, and the deletion error rate is the number of deleted bases averaged over the total aligned length of the assembly to the truth. The overall error rate is the sum of the identity, insertion, and deletion error rates. For the purpose of simplification, we used the indel error rate, which is the sum of insertion and deletion error rates.

The assess_assembly script takes an input assembly and a reference assembly to compare against. The assessment tool chunks the reference assembly to 1 Kb regions and aligns it back to the input assembly to get a trimmed reference. Next, the input is aligned to the trimmed reference sequence with the same alignment parameters to get an input assembly to the reference assembly alignment. The total aligned length is the sum of the lengths of the trimmed reference segments where the input assembly has an alignment. The total aligned length is used as the denominator while averaging each of the error categories to limit the assessment in only correctly assembled regions. Then the tool uses `stats_from_bam`, which counts the number of mismatch bases, insert bases, and delete bases at each of the aligned segments and reports the error rate by averaging them over the total aligned length.

The Pomoxis section in Supplementary Notes describe the commands we ran to perform this assessment.

Truth assemblies for base-level error-rate analysis

We used HG002, HG00733, and CHM13 for base-level error-rate assessment of the assembler and the polisher. These three assemblies have high-quality assemblies publicly available, which are used as the ground truth for comparison. Two of the samples, HG002 and HG00733, are diploid samples; hence, we picked one of the two possible haplotypes as the truth. The reported error rate of HG002 and HG00733 include some errors arising due to the zygosity of the samples. The complete hydatidiform mole sample CHM13 is a haploid human genome which is used to assess the applicability of the tools on haploid samples. We have gathered and uploaded all the files we used for assessment in one place: `https://console.cloud.google.com/storage/browser/kishwar-helen/truth_assemblies/`.

Table 4.2: The truth assembly files with download URLs.

Sample name	Region	File type	URL
HG002	Whole genome	fasta	`HG002_GRCh38_h1.fa`
		bed	`HG002_GRCh38.bed`
HG00733	Whole genome	fasta	`hg00733_truth_assembly.fa`
CHM13	Whole genome	fasta	`CHM13_truth_assembly.fa`
	Chr-X	fasta	`CHRX_CHM13_truth_assembly.fa`

To generate the HG002 truth assembly, we gathered the publicly available Genome-in-a-bottle (GIAB) high-confidence variant set (VCF) against GRCh38 reference sequence. Then we used `bedtools` to create an assembly (FASTA) file from the GRCh38 reference and the high-confidence variant set. We got two files using this process for each of the haplotypes, and we picked one randomly as the truth. All the diploid HG002 assembly is compared against this one chosen assembly. GIAB also provides a bed file annotating high-confidence regions where the called variants are highly precise and sensitive. We used this bed file with `assess_assembly` to ensure that we compare the assemblies only in the high confidence regions.

The HG00733 truth is from the publicly available phased PacBio high-quality assembly of this sample [157]. We picked phase0 as the truth assembly and acquired it from NCBI under accession `GCA_003634895.1`. We note that the assembly is phased but not haplotyped, such that portions of phase0 will include sequences from both parental haplotypes and is not suitable for trio-binned analyses. Furthermore, not all regions were fully phased; regions with variants that are represented as some combination of both haplotypes will result in lower QV and a less accurate truth.

For CHM13, we used the v0.6 release of CHM13 assembly by the T2T consortium [206]. The reported quality of this truth assembly in Q-value is QV 39. One of the attributes of this assembly is chromosome X. As reported by the T2T assembly authors, chromosome X of CHM13 is the most complete (end-to-end) and high-quality assembly of any human chromosome. We obtained the chromosome X assembly, which is the highest-quality truth assembly (QV >= 40) we have.

QUAST / BUSCO

To quantify contiguity, we primarily depended on the tool QUAST [138]. QUAST identifies misassemblies as major rearrangement events in the assembly relative to the reference. We use the phrase *disagreement* in our analysis, as we find "misassembly" inappropriate considering potentially true structural variation. For our assemblies, we quantified all contiguity stats against GRCh38, using autosomes plus chromosomes X and Y only. We report the total disagreements given that their relevant "size" descriptor was greater than 1 Kb, as is the default behavior in QUAST. QUAST provides other contiguity statistics in addition to disagreement count, notably total length and total aligned length as reported in Figure 4.2d. To determine total aligned length (and unaligned length), QUAST performs collinear chaining on each assembled contig to find the best set of non-overlapping alignments spanning the contig. This process contributes to QUAST's disagreement determination. We consider unaligned sequence to be the portions of the assembled contigs which are not part of this best set of non-overlapping alignments. All statistics are recorded in Supplementary Table B.5. For all QUAST analyses, we used the flags `min-identity 80` and `fragmented`.

QUAST also produces an NGAx plot (similar to an NGx plot) which shows the aligned segment size distribution of the assembly after accounting for disagreements and unalignable regions. The intermediate segment lengths that would allow NGAx plots to be reproduced across multiple samples on the same axis (as is shown in Figure 4.2b) are not stored, so we created a GitHub fork of QUAST to store this data during execution: `https://github.com/rlorigro/quast`. Finally, the assemblies and the output of QUAST were parsed to generate figures with an NGx visualization script, `ngx_plot.py`, found at `github.com/rlorigro/nanopore_assembly_and_polishing_assessment/`. For NGx and NGAx plots, a total genome size of 3.23Gb was used to calculate cumulative coverages.

BUSCO [197] is a tool which quantifies the number of Benchmarking Universal Single-Copy Orthologs present in an assembly. We ran BUSCO via the option within QUAST, comparing against the *eukaryota* set of orthologs from OrthoDB v9.

Disagreement assessments

To analyze the QUAST-reported disagreements for different regions of the genome, we gathered the known segmental duplication (SD) regions [24], centromeric regions for GRCh38, and known regions in GRCh38 with structural variation for HG002 from GIAB [234]. We used a Python script `quast_sv_extractor.py` that compares each reported disagreement of QUAST to the SD, SV and centromeric regions and discounts any disagreement that overlaps with these regions. The `quast_sv_extractor.py` script can be found at `https://github.com/kishwarshafin/helen/blob/master/modules/python/helper/`.

The segmental duplication regions of GRCh38 defined in the `ucsc.collapsed.sorted.segdups` file can be downloaded from `https://github.com/mvollger/segDupPlots/`.

The defined centromeric regions of GRCh38 for all chromosomes are used from the available summary at `https://www.ncbi.nlm.nih.gov/grc/human`.

For GIAB HG002, known SVs for GRCh38 are available in `NIST_SVs_Integration_v0.6/` under `ftp://ftp-trace.ncbi.nlm.nih.gov/giab/ftp/data/AshkenazimTrio/analysis/`. We used the `Tier1+2` bed file available at the GIAB ftp site.

We further exclude SV enriched regions like centromeres, secondary constriction regions, acrocentric arms, large tandem repeat arrays, segmental duplications and the Y chromosome plus 10 kbp on either side of them. The file is available here `https://github.com/kishwarshafin/helen/blob/master/masked_regions/GRCh38_masked_regions.bed`.

To analyse disagreements within the intersection of the assembled sequences we performed the following analysis. For each assembly we used `minimap2` and `samtools` to create regions of unique alignment to GRCh38. For minimap2 we used the options `-secondary=no -a -eqx -Y -x asm20 -m 10000 -z 10000,50 -r 50000 -end-bonus=100 -O 5,56 -E 4,1 -B 5`. We fed these alignments into `samtools view` with options `-F 260 -u -` and then `samtools sort` with option `-m`. We then scanned 100 basepair windows of GRCh38 to find windows where all assemblies for the given sample were aligned with a 1-1 mapping to GRCh38. We then report the sum of disagreements across these windows. The script for this analysis is here: `https://github.com/mvollger/consensus_regions`.

Trio-binning

We performed trio-binning on two samples HG002 and HG00733 [95]. For HG00733, we obtained the parental read sample accessions (HG00731, HG00732) from 1000 genome database. Then we counted k-mers with `meryl` to create maternal and paternal k-mer sets. Based on manual examination of the k-mer count histograms to determine an appropriate threshold, we excluded k-mers occurring less than 6 times for maternal set and 5 times for paternal set. We subtracted the paternal set from the maternal set to get k-mers unique to the maternal sample and similarly derived unique paternal k-mer set. Then for each read, we counted the number of occurrences of unique maternal and paternal k-mers and classified the read based on the highest occurrence count. During classification, we avoided normalization by k-mer set size. This resulted in 35.2x maternal, 37.3x paternal, and 5.6x unclassified for HG00733. For HG002, we used the Illumina data for the parental samples (HG003, HG004) from GIAB project [232]. We counted k-mers using `meryl` and derived maternal paternal sets using the same protocol. We filtered k-mers that occur less than 25 times in both maternal and paternal sets. The classification resulted in 24x maternal, 23x paternal, and 3.5x unknown. The commands and data source are detailed in the Supplementary Notes.

Transcript analysis with comparative annotation toolkit

We ran the Comparative Annotation Toolkit [55] to annotate the polished assemblies in order to analyze how well Shasta assembles transcripts and genes. Each assembly was individually aligned to the GRCh38 reference assembly using Cactus [160] to create the input alignment to CAT. The GENCODE [79] V30 annotation was used as the input gene set. CAT was run in the transMap mode only, without Augustus refinement, since the goal was only to evaluate the quality of the projected transcripts. All transcripts on chromosome Y were excluded from the analysis since some samples lacked a Y chromosome.

Run-Length Confusion Matrix

To generate run-length confusion matrices from reads and assemblies, we run-length encoded (RLE) the assembly/read sequences and reference sequences using a purpose-built python script, `measure_runlength_distribution_from_fasta.py`. The script requires a reference and sequence file, and can be found in the GitHub repo `https://github.com/rlorigro/runlength_analysis/`. The RLE nucleotides were aligned to the RLE reference nucleotides with `minimap2`. As RLE sequences cannot have identical adjacent nucleotides, the number of unique k-mers is diminished with respect to standard sequences. As `minimap2` uses empirically determined sizes for seed k-mers, we used a k-mer size of 19 to approximately match the frequency of the default size (15) used by the presets for standard sequences. For alignment of reads and assemblies we used the `map-ont` and `asm20` presets respectively.

By iterating through the alignments, each match position in the cigar string (mismatched nucleotides are discarded) was used to find a pair of lengths (x,y) such that x is a predicted length and y is the true (reference) length. For each pair, we updated a matrix which contains the frequency of every possible pairing of prediction vs truth, from length 1bp to 50bp. Finally, this matrix is normalized by dividing each element by the sum of the observations for its true run length, $\sum_{i=1}^{50}(x_i, y)$, and plotted as a heatmap. Each value represents the probability of predicting a length for a given true length.

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.

Assembly of MHC

Each of the 8 GRCh38 MHC haplotypes were aligned using `minimap2` (with preset `asm20`) to whole genome assemblies to identify spanning contigs. These contigs were then extracted from the genomic assembly and used for alignment visualization. For dot plots, Nucmer 4.0 [100] was used to align each assembler's spanning contigs to the standard chr6:28000000-34000000 MHC region, which includes 500Mb flanks. Output from this alignment was parsed with Dot [144] which has a web-based GUI for visualization. All defaults were used in both generating the input files and drawing the figures. Coverage plots were generated from reads aligned to chr6, using a script, `find_coverage.py`, located at (`github.com/rlorigro/nanopore_assembly_and_polishing_assessment/`).

The best matching alt haplotype (to Shasta, Canu, and Flye) was chosen as a reference haplotype for quantitative analysis. Haplotypes with the fewest supplementary alignments across assemblers were top candidates for QUAST analysis. Candidates with comparable alignments were differentiated by identity. The highest contiguity/identity MHC haplotype was then analyzed with QUAST using `-min-identity 80`. For all MHC analyses regarding Flye, the unpolished output was used.

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 [98, 214] and 179 for HG00733 [24] (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`.

Short Read Polishing

Chromosome X of the CHM13 assembly (assembled first with Shasta, then polished with MarginPolish and HELEN) was obtained by aligning the assembly to GRCh38 (using `minimap2` with the `-x asm20` flag). 10X Chromium reads were downloaded from the Nanopore WGS Consortium (`https://github.com/nanopore-wgs-consortium/CHM13/`). These were from a NovaSeq instrument at a coverage of approximately 50X. The reads corresponding to chromosome X were extracted by aligning the entire read set to the whole CHM13 assembly using the 10X Genomics `Long Ranger Align` pipeline (v2.2), then extracting those corresponding to the corresponding chromosome X contigs with `samtools`. Pilon [218] was run iteratively for a total of three rounds, in each round aligning the reads to the current assembly with `Long Ranger` and then running Pilon with default parameters.

Structural Variant Assessment

To create an assembly graph in GFA format Shasta v0.1.0 was run using the HG002 sequence data with `-MarkerGraph.simplifyMaxLength 10` to reduce bubble removal and `-MarkerGraph.highCoverageThreshold 10` to reduce the removal of edges normally removed by the transitive reduction step.

To detect structural variation inside the assembly graphs produced by Shasta, we extracted unitigs from the graph and aligned them back to the linear reference. Unitigs are walks through the assembly graph that do not traverse any node end that includes a bifurcation. We first processed the Shasta assembly graphs (in GFA format) with `gimbricate` (`https://github.com/ekg/gimbricate c1c6d1a`) to recompute overlaps in non run-length encoded space and to remove nodes in the graph only supported by a single sequencing read. To remove overlaps from the graph edges, we then "bluntified" resulting GFAs with `vg find -F` (`https://github.com/vgteam/vg v1.19.0 Tramutola`). We then applied `odgi unitig` (`https://github.com/vgteam/odgi 463ba5b`) to extract unitigs from the graph, with the condition that the starting node in the unitig generation must be at least 100 bp long. To ensure that the unitigs could be mapped back to the linear reference, we appended a random walk of 25 Kb after the natural end of each unitig, with the expectation that even should unitigs would yield around 50 Kb of mappable sequence. Finally, we mapped the unitigs to GRCh38 with `minimap2` with a bandwidth of 25 Kb (`-r25000`), and called variants in the alignments using `paftools.js` from the `minimap2` distribution. We implemented the process in a single script that produces variant calls from the unitig set of a given graph `https://github.com/ekg/shastaGFA/blob/master/shastaGFAtoVCF_unitig_paftools.sh`.

The extracted variants were compared to the structural variants from the Genome In A Bottle benchmark in HG002 (v0.6, [235]). Precision, recall and F1 scores were computed on variants not overlapping simple repeats and within the benchmark's high-confidence regions. Deletions in the assembly and the GIAB benchmark were matched if they had at least 50% reciprocal overlap. Insertions were matched if located at less than 100 bp from each other and similar in size (50% reciprocal similarity).

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 [104] (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 [210], 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 [46], 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 [28]. Most earlier algorithms for constructing POA graphs have a well known explicit read order dependence that can result in undesirable topologies [104].

- MarginPolish works in run-length encoded space, which results in considerably less alignment uncertainty and correspondingly improved performance.

- MarginPolish, similarly to Nanopolish [121], 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 [160]. 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 [87] 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 [161]. 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:

$$w(j,x) = \sum_{r \in R} \sum_{i} \mathbb{1}_x(r_i) P(r_i \diamond s_j)$$

where $\mathbb{1}_x(r_i)$ 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 $j, j+1, \ldots k-1$ 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).

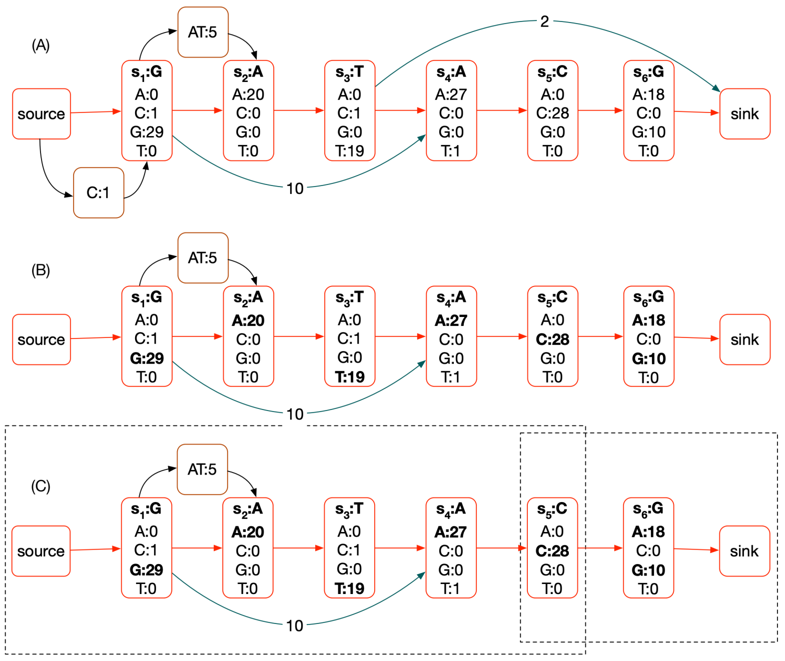


Figure 4.6: A) An example POA, assuming approximately 30x read coverage. The backbone is shown in red. Each non-source/sink node has a vector of weights, one for each possible base. Deletion edges are shown in teal, they also each have a weight. Finally insertion nodes are shown in brown, each also has 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.

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.

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.

A Simple Bayesian Model for Run-length Decoding

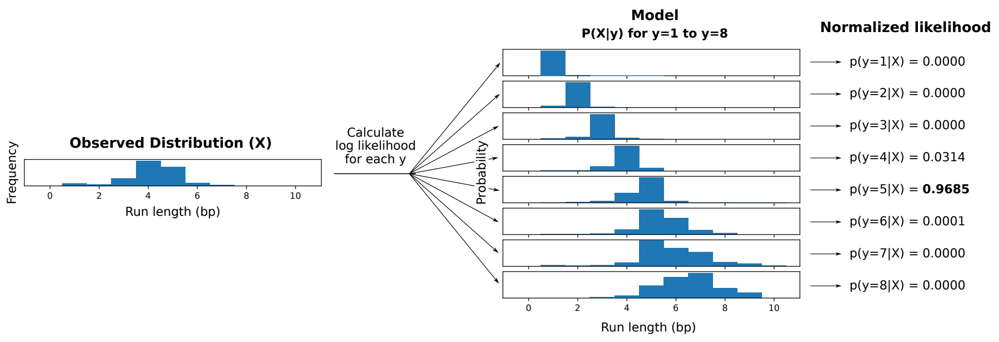


Figure 4.7: **Visual representation of run length inference.** This diagram shows how a consensus run length is inferred for a set of aligned lengths (X) that pertain to a single position. The lengths are factored and then iterated over, and log likelihood is calculated for every possible true length 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.

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).

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 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. 4.10).

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 CITE:PMID: 28398314.

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.

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) [131] 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 a 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.

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.

a.

(i)
Assembly sequence: GGAAAAAAAAACATTTTAAAA
True sequence: GGAAAAAAAA - - TTTTAAAA

Assembly sequence in run-length:	G	A	A	C	A	T	A
	2	5	3	1	1	4	4
Truth sequence in run-length:	G	A	A	-	-	T	A
	2	5	3	0	0	4	4

(ii)
Assembly sequence: ATGAAA - - CTTG
True sequence: ATGAAAGGCTTG

Assembly sequence in run-length:	A	T	G	A		C	T	G
	1	1	1	3		1	2	1
Truth sequence in run-length:	A	T	G	A	G	C	T	G
	1	1	1	3	2	1	2	1

b.

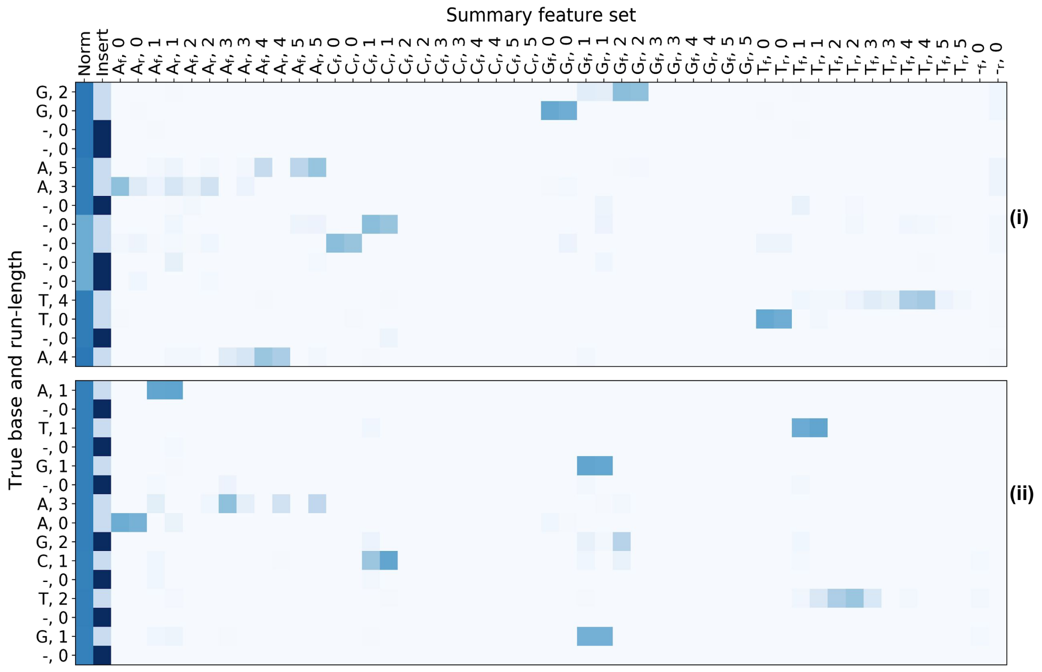


Figure 4.8: **MarginPolish Images** A graphical representation of images from two labeled regions selected to demonstrate: the encoding of a single POA node into two run-length blocks (i), a true deletion (i), and a true insert (ii). The y-axis shows truth labels for nucleotides and run-lengths, the x-axis describes features in the images, and colors show associated weights.

The 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.

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. 4.8 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 Margin-Polish (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.

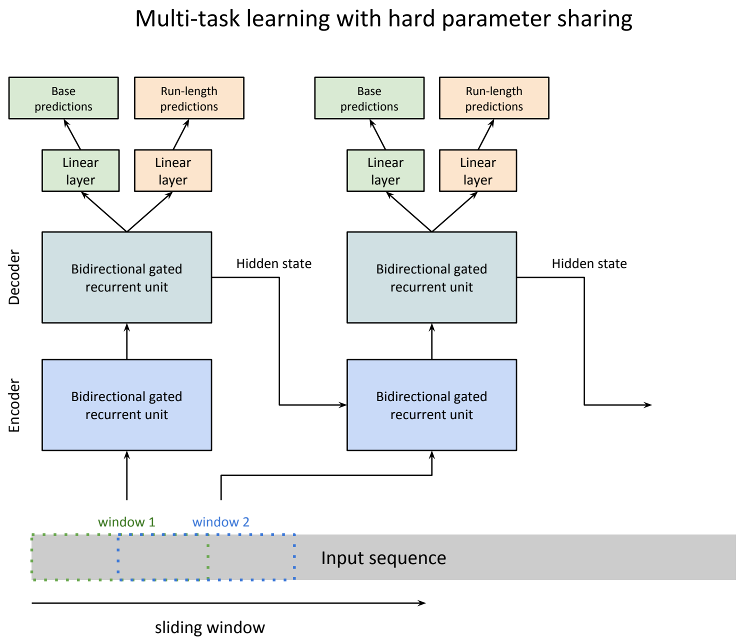


Figure 4.9: The sequence-to-sequence model implemented in Helen.

The model

We use a sequence transduction model for consensus polishing. The model structure consists of two single-layer gated recurrent neural units (GRU) for encoding and decoding on top of two linear transformation layers. The two linear transformation layers independently predict a base and a run-length for each position in the input sequence.

Each unit of the GRU can be described using the four functions it calculates:

$$\begin{gathered} r_t = Sigmoid(W_{ir}x_t + W_{hr}h_{(t-1)}) \\ u_t = Sigmoid(W_{iu}x_t + W_{hu}h_{(t-1)}) \\ n_t = tanh(W_{in}x_t + r_t * (W_{hn}h_{(t-1)})) \\ h_t = (1 - u_t) * n_t + u_t * h_{(t-1)} \end{gathered} \tag{4.1}$$

For each genomic position t, we calculate the current state h_t from the new state n_t and the update value u_t applied to the output state of previous genomic position $h_{(t-1)}$. The update function u_t decides how much past information to propagate to the next genomic position. It multiplies the input x_t with the weight vector W_{iu} and multiplies the hidden state of the previous genomic position $h_{(t-1)}$. The weight vectors decide how much from the previous state to propagate to the next state. The reset function r_t decides how much information to dissolve from the previous state. Using a different weight vector, the r_t function decides how much information to dissolve from the past. The new memory state n_t is calculated by multiplying the input x_t with the weight vector W_{in} and applying a Hadamard multiplication $*$ between the reset function value and a weighted state of the previous hidden state $h_{(t-1)}$. The new state captures the associative relationship between the input weights and true prediction. In this setup, we can see that r_t and u_t can decide to hold memory from distant locations while n_t captures the associative nature of the weights to the prediction, helping the model to decide how to propagate genomic information in the sequence. The output of each genomic position h_t can be then fed to the next genomic position as a reference to the previously decoded genomic position. The final two layers apply linear transformation functions:

$$\begin{aligned} B_t &= h_t * W^T \\ R_t &= h_t * W^T \end{aligned} \tag{4.2}$$

The two linear transformation functions independently calculate a base prediction B_t and a run-length prediction R_t from the hidden state output of that genomic position h_t. The model operates in hard parameter sharing mode where the model learns to perform two tasks in equation 4.2 using the same set of underlying parameters from equation 4.1. The ability of the model to reduce the error rate of the assemblies from multiple samples with multiple assemblers shows the generalizability and robustness we achieve with this method.

Sliding window mechanism

One of the challenges of this setup is the sequence length. From the functions of recurrent units in equation 4.1, we see that each state is updated based on the previous state and associated weight. Due to the noisy nature of the data, if the sequence length is too long, the back-propagation becomes difficult over noisy regions. On the other hand, a small sequence length would make the program very slow. We balance the run-time and accuracy by using a sliding window approach.

During the sliding-window, we chunk the sequence of thousand bases to multiple overlapping windows of length 100. Starting from the leftmost window, we perform prediction on sequence pileups of the window and transmit the hidden state of the current window to the next window and slide the window by 50 bases to the right. For each window, we collect all the predicted values and add it to a global sequence inference counter that can keep track of predicted probabilities of base and run-length at each position. Lastly, we aggregate the probabilities from the global inference counter to generate a sequence. This setup allows us to utilize the minibatch feature of the popular neural network libraries allowing inference on a batch of inputs instead of performing inference one at a time.

Training the model

HELEN is trained with a gradient descent method. We use Adaptive Moment Estimation (Adam) method to compute gradients for each of the parameters in the model based on a target loss function. Adam uses both decaying squared gradients and the decaying average of gradients, making it suitable to use with recurrent neural networks[131]. Adam performs gradient optimization by adapting the parameters to set in a way that minimizes the value of the loss function.

We perform optimization through back-propagation per window of the input sequence. From equation 4.2, we see that we get two vectors $B = [B_1, B_2, B_3...B_n]$ and $R = [R_1, R_2, R_3...R_n]$ containing base and run-length predictions for each window of size n.

From the labeled data we get two more such vectors $T_B = [T_{B1}, T_{B2}, T_{B3}, ...T_{Bn}]$ and $T_R = [T_{R1}, T_{R2}, T_{R3}, ...T_{Rn}]$ containing the true base and true rle values of each position in the window. From these loss function the loss L is calculated:

$$\begin{aligned} L_B(B, T_B) &= -B[T_B] + \log\left(\sum_j \exp(B[j])\right) \\ L_R(R, T_R) &= weight[T_R]\left(-R[T_R] + \log\left(\sum_j \exp(R[j])\right)\right) \\ L &= L_B + L_R \end{aligned} \tag{4.3}$$

In equation 4.3, L_B calculates the base prediction loss and L_R calculates the rle prediction loss. The rle class distribution is heavily biased toward lower run-length values, so, we apply class-wise weights depending on the observation of per class to make the learning process balanced between classes. The optimizer then updates the parameters or weights W of the model from equation 4.1 and equation 4.2 in a way that minimizes the value of the loss function. We can see that the loss function is a summation of the two independent loss functions but the underlying weights from the recurrent neural network belongs to the same set of elements in the model. In this setting, the model optimizes to learn both task simultaneously by updating the same set of weights.

Sequence stitching

To parallelize the polishing pipeline, MarginPolish chunks the genome into smaller segments while generating images. Each image segment encodes a thousand nucleotide bases, and two adjacent chunks have 50 nucleotide bases overlap between them. During the inference step, we save all run-length and base predictions of the images, including their start and end genomic positions.

For stitching, we load all the image predictions and sort them based on the genomic start position of the image chunk and stitch them in parallel processes. For example, if there are n predictions from n images of a contig and we have t available threads, we divide n prediction chunks into t buckets each containing approximately n/t predicted sequences. Then we start t processes in parallel where each process stitches all the sequences assigned to it and returns a longer sequence. For stitching two adjacent sequences, we take the overlapping sequences between the two sequence and perform a pairwise Smith-Waterman alignment. From the alignment, we pick an anchor position where both sequences agree the most and create one sequence. After all the processes finish stitching the buckets, we get t longer sequences generated by each process. Finally, we iteratively stitch the t sequences using the same process and get one contiguous sequence for the contig.

Generating trained models

In supplementary tables B.20, B.23 and B.22 we report several models for HELEN. The models are trained on different sets of data with varying Guppy base-caller versions. We discuss three trained models `r941_flip235_v001.pkl`, `r941_flip233_v001.pkl`, and `r941_flip231_v001.pkl` to use with HELEN for different versions of the ONT Guppy base-callers. Due to the difference in the error profile of different versions of the Guppy base-caller, we trained three different models.

Table 4.3: Description of trained models for HELEN.

Model Name	Base caller version	Training sample	Training region	Testing region
r941_flip235_v001.pkl	Guppy 2.3.5	HG002	Chr1-19, Chr21-22	Chr20
r941_flip233_v001.pkl	Guppy 2.3.3	HG002	Chr1-19, Chr21-22	Chr20
r941_flip231_v001.pkl	Guppy 2.3.1	CHM13	Chr1-6	Chr20

The `r941_flip235_v001.pkl` is trained on HG002 base called with Guppy 2.3.5. The model is trained on the high confidence regions of all autosomes and tested on Chr20. The training script trained the model for 80 hours on 10 epochs, which generated 10 trained models. We picked the model that has the best performance on Chr20 as the final model.

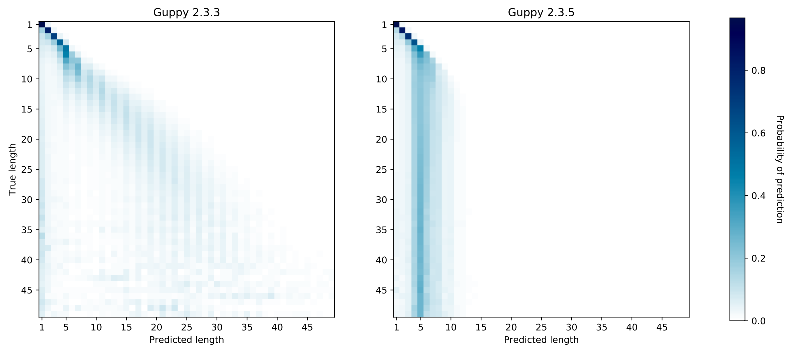


Figure 4.10: Run-length confusion in different versions of Guppy base caller

The CHM13 data from T2T consortium [206] were base called with Guppy 2.3.1. The error profile of Guppy 2.3.1 is significantly different than Guppy 2.3.5. Figure 4.10 shows the difference in underlying error profile of HG00733 sample for two different versions of Guppy. We trained `r941_flip233_v001.pkl` Model on HG002 Guppy 2.3.3 data. Although the error profile of Guppy 2.3.1 and Guppy 2.3.3 are similar, the reported base qualities are different. So, we trained another model `r941_flip231_v001.pkl` on Chr1-6 of CHM13 to see further improvement in the consensus quality of CHM13.

Implementation notes

We have implemented HELEN using python and C++ programming language. We use PyTorch [159] deep neural network library for the model implementation. We also use the Striped-Smith Waterman algorithm implementation to use during stitching and Pybind11 [91] as a bridge between C++ and python methods. The image data is saved using HDF5 file format. The implementation is publicly available via GitHub (`https://github.com/kishwarshafin/helen`).

Part V

Validation and polishing of the first complete human genome.

Chapter 5

Validation and polishing strategies for telomere-to-telomere genome assemblies

Preamble

This chapter contains the text from a pre-print titled "Chasing perfection: validation and polishing strategies for telomere-to-telomere genome assemblies"[128]. The manuscript details a pipeline we used to validate and polish the first telomere-to-telomere assembly of a human genome as a part of the T2T consortium led by Adam M Phillippy and Karen Miga. I am a co-first author of this manuscript with Ann McCartney and Michael Alonge, and Arang Rhie led the work. My contribution to this manuscript is the small variant derivation and telomere polishing with PEPPER.

I believe this is a significant contribution to the field of genomics that shows how platform-specific biases can affect even the most accurate assembly we can produce and how to correct them. I thank the T2T consortium, Arang Rhie, Ann McCartney, and Michael Alonge, for allowing me to be a part of the polishing team and share this work as a part of my dissertation. I would also like to thank Andrey V Bzikadze, Giulio Formenti, Arkarachai Fungtammasan, Kerstin Howe, Chirag Jain, Sergey Koren, Glennis A Logsdon, Karen H Miga, Alla Mikheenko, Benedict Paten, Alaina Shumate, Daniela C Soto, Ivan Sovic, Jonathan MD Wood, Justin M Zook who are co-authors of this manuscript.

Introduction

Genome assembly is a foundational practice of quantitative biological research with increasing utility. By representing the genomic sequence of a sample of interest, genome assemblies enable researchers to annotate important features, quantify functional data, and discover/genotype genetic variants in a population[147, 212, 67, 45, 83, 1]. Modern draft eukaryotic genome assembly graphs are typically built from a subset of four Whole Genome Shotgun (WGS) sequencing data types: Illumina short reads[208, 134], Oxford Nanopore Technologies (ONT) long reads[118], PacBio Continuous Long Reads (CLR), and PacBio High-Fidelity (HiFi) long reads[223, 118], all of which have been extensively described[223, 118, 208, 134]. However, we note that even the high-accuracy technologies produce sequencing data with some noise caused by platform-specific technical biases that require careful validation and polishing[9, 77, 25, 223, 147].

Current genome assembly software attempts to reconstruct an individual or mosaic haplotype sequence from a subset of the above WGS data types. Some assemblers do not attempt to correct sequencing errors[109], while others attempt to remove errors at various stages of the assembly process[94, 150, 27, 230, 195]. Regardless, technology-specific sequencing errors usually lead to distinct assembly errors[25, 219]. Additionally, suboptimal assembly of specific loci often causes small and large errors in draft assemblies[184, 176]. Here, we define "polishing" as the process of removing these errors from draft genome assemblies. Most polishing tools use an approach that is similar to sequence-based genetic variant discovery. Specifically, reads from the same individual are aligned to a draft assembly, and putative "variant"-like sequence edits are identified[176, 231]. For diploid genomes, heterozygous "alternate" alleles are interpreted as genuine heterozygous variants, while homozygous alternate alleles are interpreted as assembly errors to be corrected. Some polishing tools, such as Quiver/Arrow, Nanopolish, Medaka, DeepVariant, and PEPPER leverage specialized models and prior knowledge to correct errors caused by technology-specific bias[158, 122, 167, 191, 156]. Others, such as Racon[209], use generic methods to correct assembly errors with a subset of sequencing technologies[209, 229, 61]. These generic tools can utilize multiple data types to synergistically overcome technology-specific assembly errors.

The Telomere-to-Telomere (T2T) consortium recently convened an international workshop to assemble the first-ever complete sequence of a human genome. Because heterozygosity can complicate assembly algorithms, the consortium chose to assemble the highly homozygous genome of a complete hydatidiform mole cell line (CHM13hTERT; abbr. CHM13). Primarily using HiFi reads and supplemented with ONT reads, the consortium built a highly accurate and complete draft assembly (CHM13v0.9) that resolved all repeats with the exception of the rDNAs[147]. CHM13v0.9 contained about 1 error in every 10.5 Mb (Q70.22), and while this was highly accurate by traditional standards, we, as part of the consortium, sought to correct all lingering errors and omissions, including those within repeats, in this first truly complete assembly of a human genome.

Alignment-based validation and polishing commonly underperform within genomic repeats where alignments are ambiguous and inaccurate. For example, this challenge was identified while validating the first complete centromere and satellite repeats of the Chromosome X, requiring a customized conservative marker-assisted alignment[4]. To address this challenge, specialized repeat-aware alignment methods were recently developed, such as Winnowmap2[86, 84] and TandemMapper[137]. However, to the best of our knowledge, no studies have utilized such methods to reliably validate and polish an entire genome assembly, including the most notoriously repetitive regions.

Here, we describe techniques developed to carefully evaluate the accuracy and completeness of a complete human genome assembly using multiple complementary WGS data types. Our evaluation of the initial draft CHM13 assembly discovered a number of assembly errors, therefore we created a custom polishing pipeline that was robust to genomic repeats and technology-specific biases. By applying this polishing pipeline to

CHM13v0.9, we made 1,457 corrections, replacing a total of 12,234,603 bp of sequence with 10,152,653 bp of sequence, ultimately leading to the landmark CHM13v1.1 assembly representing the first complete human genome ever assembled. Our edits increased the estimated quality value to Q73.94 while mitigating haplotype switches. Further, we extended the truncated p-arm of chromosome 18 to encompass the complete telomere, and polished all telomeres with a new specialized PEPPER-DeepVariant model. Our careful evaluation of CHM13v1.1 confirmed that polishing did not overcorrect repeats (including rDNAs) nor did it cause false-positive edits causing invalid coding sequence reading frames. Additionally, we identified a comprehensive list of putatively heterozygous loci in the CHM13 cell line, as well as sporadic loci where read alignments still indicated exceptionally low coverage. Finally, we uncovered common mistakes made by standard automated polishing pipelines and provide best practices for other genome assembly projects.

Results

Initial evaluation of CHM13v0.9

The T2T Consortium has collected a comprehensive and diverse set of publicly available WGS sequencing and genomic map data (Illumina PCR-free, PacBio HiFi, PacBio CLR, ONT, and Bionano optical maps) for the nearly-completely homozygous CHM13 cell line (https://github.com/nanopore-wgs-consortium/CHM13). As part of the consortium, we drew upon these sequencing data to generate a custom pipeline (Figure 5.1) to evaluate, identify and correct lingering errors in CHM13v0.9.

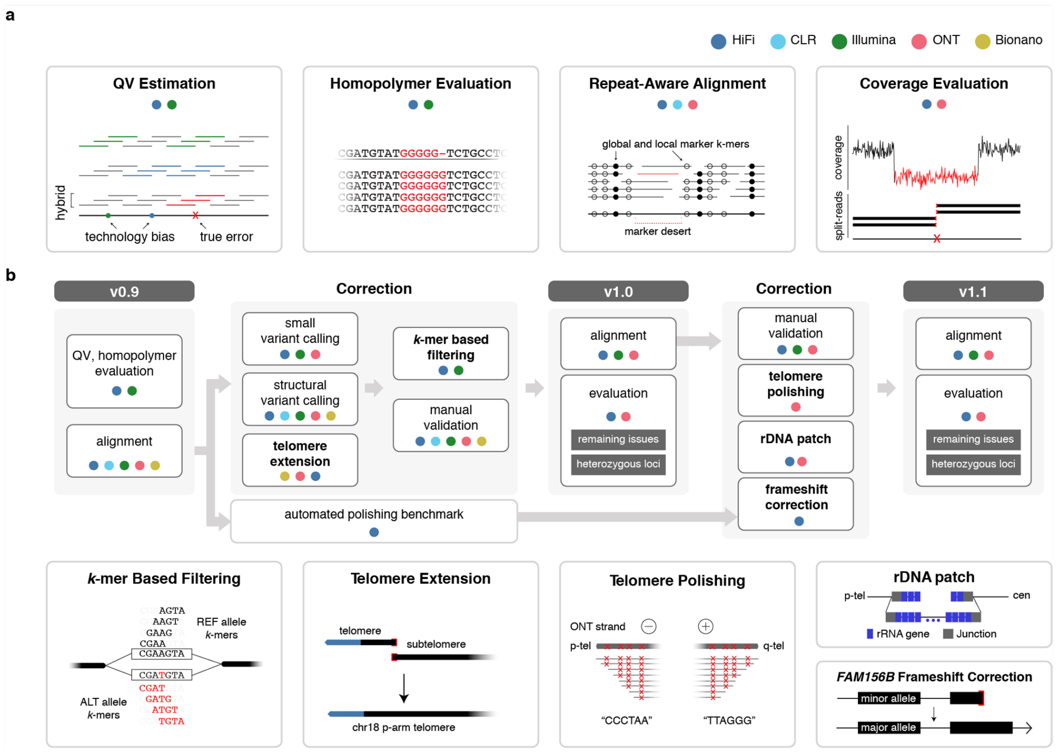


Figure 5.1: An overview of the evaluation and polishing strategy developed to achieve a complete , polished, human genome. a. The evaluation strategies applied to assess consensus genome assembly accuracy both before (CHM13v0.9) and after (CHM13v1.0 and CHM13v1.1). b. The "do no harm" polishing strategy developed and implemented to generate CHM13v1.0 and CHM13v1.1 after the initial evaluation of the CHM13v0.9 consensus assembly.

We first derived k-mer-based quality estimations ($k = 21$bp) of CHM13v0.9 using Merqury[177] using both Illumina and HiFi reads. The k-mer size was chosen to limit the collision rate to 0.5% given the estimated genome size of 3.05 Gbp of CHM13[57]. While estimating the Illumina reads QV, we found 15,723 k-mers present in the assembly and not the reads (erroneous k-mers), leading to an estimated base quality of Q66.09. Using HiFi reads, we found 6,881 error k-mers (Q69.68) (Figure 5.2). To test how technical sequencing bias may have influenced this QV estimation, we examined the k-mer multiplicity and sequence content of assembly k-mers absent from one technology but present in the other. Here, our results indicated that k-mers missing from Illumina reads were present with expected frequency in HiFi and were enriched for G/C bases. Conversely, k-mers missing in HiFi were present with higher frequency in Illumina reads with A/T base enrichment (Figure 5.2b). However, we identified no particular enrichment pattern in the number of GA or CTs within the k-mers, possibly due to the short k-mer size chosen (Supplementary Figure C.1a). Most of the k-mers absent from HiFi reads were located in patches derived from a previous ONT-based assembly (CHM13v0.7), which were included to overcome regions of HiFi coverage dropout[147] (Supplementary figure C.1b-c). These findings highlighted that platform-specific sequencing biases were underestimating the QV when measured from a single sequencing platform. To overcome this, we created a hybrid k-mer database that combined these platforms to be used for QV estimation (Supplementary figure C.1d). Unlike the default QV estimation in Merqury, we removed low frequency k-mers to avoid overestimated QVs caused by excessive noise accumulated from both platforms. We estimated base level accuracy as Q70.22 with 6,073 missing k-mers (Supplementary table C.1). We note that this estimate does not account for the rarer case of k-mers present in the reads but misplaced or falsely duplicated in the assembly.

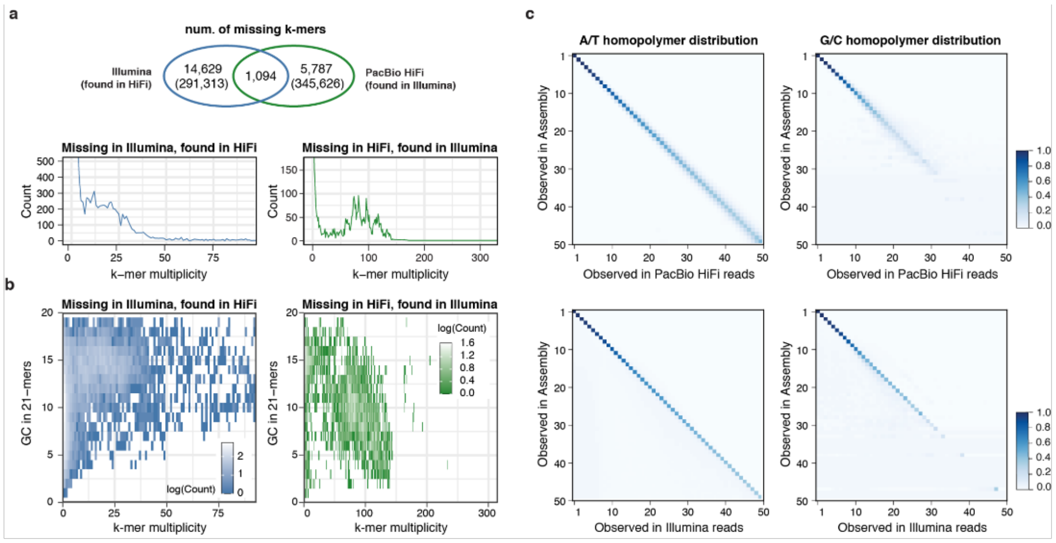


Figure 5.2: Sequencing biases in PacBio HiFi and Illumina reads. a. Venn Diagram of the distinct "error" k-mers found only in the assembly and not in the HiFi reads (blue) or Illumina reads (green). Except the 1,085 k-mers that did not exist in either HiFi or Illumina reads, error k-mers were found in the other sequencing platform with expected frequency, matching the average sequencing coverage (lower panels). b. Missing k-mers from a with its GC contents, colored by the frequency observed. Low frequency erroneous k-mers did not have a clear GC bias. K-mers found only in HiFi had a higher GC percentage, while higher frequency k-mers tend to have more AT rich sequences in Illumina. c. Homopolymer length distribution observed in the assembly and in HiFi reads (upper) or Illumina reads (lower) aligned to that position. The longer the homopolymer length became in the consensus, the length became variable in HiFi reads especially in the GC homopolymers. Majority of the Illumina reads were continuously concordant with the consensus.

Despite the high accuracy of CHM13v0.9 (Q70.22), we expected to find consensus sequence errors related to the systematic presence of homopolymer- or repeat-specific issues in HiFi reads[118, 102]. To detect these, we generated self-alignments by aligning CHM13 reads to CHM13v0.9 for each WGS sequencing technology. Though each data type required technology-specific alignment methods, we highlight our use of Winnowmap2 that enabled robust alignment of long-reads to both repetitive and non-repetitive regions of CHM13v0.9[86, 84]. To understand the homopolymer length differences between the assembly and the reads, we derived a confusion matrix from Illumina read alignments showing discordant representation of long homopolymers between the Illumina reads and the assembly (Figure 5.2c). Altogether, the QV and homopolymer analysis suggested that CHM13v0.9 required polishing to maximize accuracy of a complete human genome.

Identification and correction of assembly errors

To address assembly flaws identified during evaluation, we aimed to establish a customized polishing pipeline that would avoid false positive polishing edits (especially in repeats) and maintain local haplotype consistency (Figure 5.1b) (Supplementary Figure C.2). We identified and corrected small errors ($<=$50bp) using several small variant calling tools from self-alignments of Illumina, HiFi, and ONT reads to CHM13v0.9. To call both single-nucleotide polymorphisms (SNPs) and small insertions and deletions (INDELs), we applied a hybrid mode of DeepVariant[167] that exploited both HiFi and Illumina read alignments[154]. Simultaneously, we used PEPPER-DeepVariant[191] to generate additional SNP calls with ONT reads as it can yield high-quality SNP variants in difficult regions of the genome[154] (Supplementary Figure C.3). We rigorously filtered all calls using Genotype Quality ($GQ < 30$ for the hybrid calls and $GQ < 25$ for ONT SNP calls) and Variant Allele Frequency ($VAF < 0.5$) to exclude any low-frequency false-positive calls (Supplementary Figure C.2). We chose $VAF < 0.5$ to avoid including heterozygous variants and the GQ threshold was chosen based on the previously reported calibration plot of DeepVariant that shows that calls that have quality above 25 or 30 are highly unlikely to result in false positives[191, 167]. We then filtered all of the suggested alternate corrections with Merfin[58], a tool concurrently developed by members of the T2T consortium, to avoid introducing error *k*-mers (Figure 5.1b, Figure 5.3c). Finally, we ignored variants near the distal or proximal rDNA junctions on the short arms of the acrocentric chromosomes to avoid homogenizing the alleles from the un-assembled rDNAs. After merging all variant calls, we identified 993 small variants (<=50bp) that represented potential assembly errors and heterozygous sites. From these 993 assembly edits, about two-thirds were homopolymer corrections (512) or low-complexity micro-satellite repeats composed of 2 distinct bases in homopolymer-compressed space (hereby noted as “2-mer”) consistent with prior observations of HiFi sequence errors or bias[150]. Across all 617 loci, we evaluated the edit distribution using both Illumina and HiFi reads and found that the majority of Illumina reads supported the longer homopolymer or 2-mer repeat lengths compared to HiFi reads, thereby uncovering systemic biases in both homopolymer and 2-mer length in HiFi reads[150] that caused the propagation of these errors into the consensus assembly sequence (Figure 5.3d).

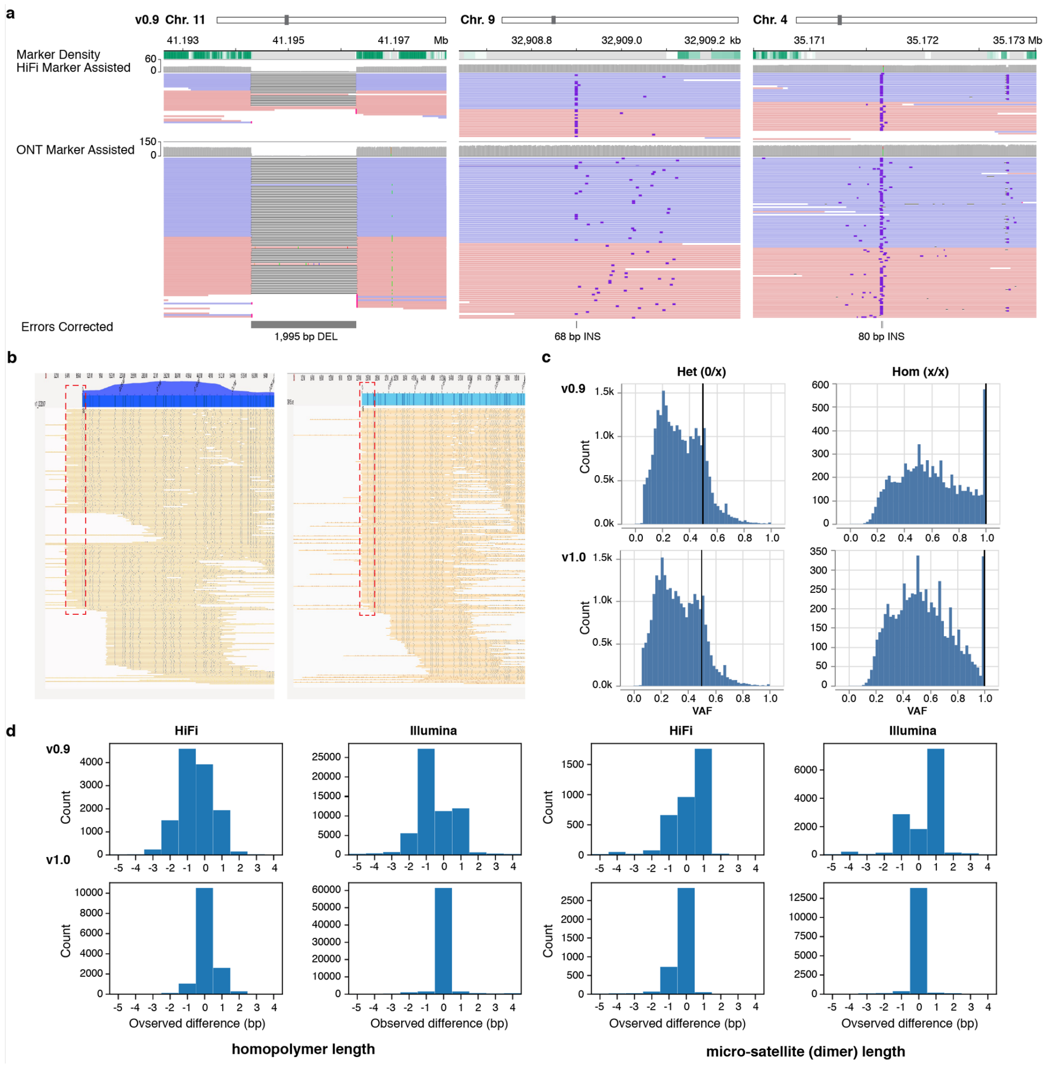


Figure 5.3: Errors corrected after polishing. a. Three SV-like errors corrected. b. Bionano optical maps indicating the missing telomeric sequence on Chr. 18 p-arm (left) with a higher than average mapping coverage. This excessive coverage were removed after adding the missing telomeric sequence (right) and most of the bionano molecules end at the end of the sequence. c. Variant allele frequency (VAF) of each variant called by DeepVariant hybrid (HiFi + Illumina) mode, before and after polishing. Most of the high frequency variants (errors) are removed after polishing, which were called as 'Homozygous' variants. d. Total number of reads in each observed length difference (bp) between the assembly and the aligned reads at each edit positions. Positive numbers indicate more bases are found in the reads, while negative number indicates less numbers in the reads. Both homopolymer and micro-satellite (dimers in homopolymer compressed space) length difference became 0 after polishing.

We used Parliament2[228] and Sniffles[188] to identify medium-sized (>50bp) assembly errors and heterozygous structural variants (SVs). Parliament2 runs 6 structural variant callers[228] using short-read data, while the Sniffles detects structural variants using one of the long-read technologies (HiFi, ONT, and CLR). To improve specificity, we only considered Parliament2 calls supported by at least two SV callers and Sniffles calls supported by at least two long-read technologies. Similar to small variant detection, we excluded SVs called in the partial rDNA arrays and the HSat3 satellite repeat on chromosome 9. This pipeline identified a relatively small number of SV calls (66, see Supplementary Figure C.2) that we were able to manually curate via genome browsing. In total, we corrected three medium-sized assembly errors (replacing 1,998 bp of CHM13v0.9 sequence with 151 bp of new sequence) and we identified 44 heterozygous SVs (Figure 5.3a and Supplementary Figure C.4). We also identified a missing telomere sequence on the p-arm of chromosome 18 — a potential result of the string graph simplification process and confirmed through Bionano mapping (Figure 5.1b, Figure 5.3b). To correct this omission, we used the CHM13v0.9 graph to identify a set of HiFi reads expected to cover this locus[147] and found ONT reads that mapped to the corresponding subtelomere and contained telomeric repeats. We used the ONT reads to derive a consensus chromosome 18 extension that was subsequently polished with the associated HiFi reads. After patching this telomere extension, we used Bionano alignments to confirm the accuracy of this locus (Figure 5.3b). Altogether, the small and medium-sized variant calls along with the chromosome 18 telomere patch were combined into two distinct VCF files: a polishing edits file (homozygous ALT variants and the telomere patch) and a file for heterozygous variants (all other variants). We created the polished CHM13v1.0 assembly by incorporating these edits into the CHM13v0.9 with bcftools[114].

We ensured polishing accuracy by extensive manual validation through visual inspection of the repeat-aware alignments, error *k*-mers, marker *k*-mers, and marker-assisted alignments. Here, we define "marker" *k*-mers as *k*-mers that occur only once in the assembly and in the expected single-copy coverage range of the read *k*-mer database and are highly likely to represent unique regions of the assembly (Supplementary Figure C.5)[136]. To generate marker-assisted alignments, we filtered Winnowmap2[86] alignments to exclude any alignments that did not span marker *k*-mers (https://github.com/arangrhie/T2T-Polish/tree/master/marker_assisted). Our findings supported that most genomic loci contained a deep coverage of marker *k*-mers to facilitate marker-assisted alignment, except for a few highly repetitive regions (11.3 Mb in total) that lacked markers (termed "marker deserts") (Figure 5.1a and Supplementary Figure C.5). In parallel, we used TandemMapper[137] to detect structural errors in all centromeric regions, including identified marker deserts. TandemMapper[137] used locally unique markers for the detection of marker order and orientation discrepancies between the assembly and associated long reads. We manually validated all large polishing edits and heterozygous SVs, and many small loci were validated *ad hoc*.

Validation of CHM13v1.0

Given the high completeness and accuracy standards of the T2T consortium, and knowing that polishing may introduce additional errors[58], we took extra precautions to validate polishing edits and to ensure that edits did not degrade the quality of CHM13v0.9. First, we repeated self-alignment variant calling methods on CHM13v1.0, confirming that all edits made were correct (Figure 5.3a). Through Bionano optical map alignments, we validated the structural accuracy of the chromosome 18 telomere patch and confirmed that all 46 telomeres were represented in CHM13v1.0 (Figure 5.3b). Notably, our polishing led to a marked improvement in the distribution of GQ and VAF of small variant calls (Figure 5.3c and Supplementary Figure C.6a). Our approach also increased the base level consensus accuracy from Q70.22 in CHM13v0.9 to Q72.62 in CHM13v1.0. Further, we found that error *k*-mers were uniformly distributed along each chromosome, suggesting that remaining errors were not clustered within certain genomic regions (Supplementary Figure C.6b-c). Upon re-evaluation of the homopolymers and 2-mers, we noted most of the biases we found in CHM13v0.9 from HiFi reads had been accurately removed, achieving an improved concordance with Illumina reads (Figure 5.3d). Polishing did not induce invalid open reading frames (ORFs) in CHM13v0.9 transcripts with valid ORFs, and polishing corrected 16 invalid CHM13v0.9 ORFs (Supplementary Table C.2).

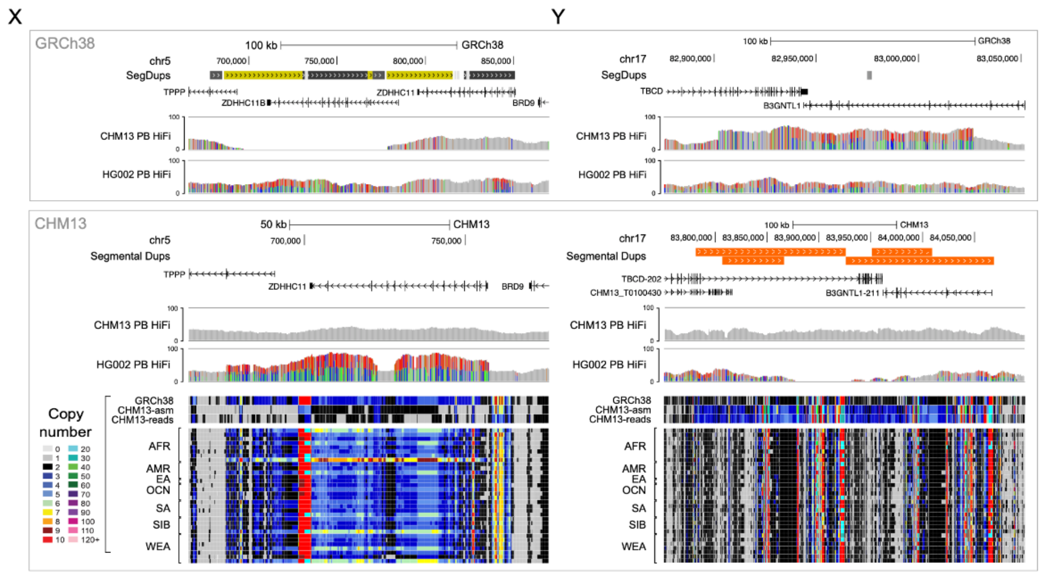


Figure 5.4: Examples of the largest CHM13 regions with a copy number in the reference that differs from GRCh38 and most individuals. a. One of the two largest examples of rare collapses in CHM13, where one copy of a common 72 kb tandem duplication is absent in CHM13. b. The largest rare duplication in CHM13, a 142 kb tandem duplication of sequence in GRCh38 that is rare in the population. CHM13 and HG002 PacBio HiFi coverage tracks are displayed for both references, GRCh38 (top) and CHM13v1.0 (bottom), to demonstrate that CHM13 reads support the CHM13 copy-number but HG002 reads are consistent with the GRCh38 copy-number. Read-depth copy-number estimates in CHM13 are shown at the bottom for 'k-merized' versions of GRCh38 and CHM13v1.0 references, CHM13 Illumina reads, and Illumina reads from a diverse subset (n=34) of SGDP individuals.

Overall, we made a total of 112 polishing edits (impacting 267 bp) in centromeric regions[4], with 15 (35 bp) of these edits occurring specifically in centromeric alpha-satellite higher-order repeat arrays. We made 134 edits (4,975 bp) in non-satellite segmental duplications[213]. Moreover, the polishing edits were neither enriched nor depleted in satellite repeats and segmental duplications (p=0.85, permutation test), suggesting that non-masked repeats were not over- or under-corrected compared to the rest of the genome (Supplementary Figure C.7). Finally, through extensive manual inspection, we confirmed the reliability of the alignments for the three SV associated edits incorporated into CHM13v1.0 (Supplementary Figure C.8), and these efforts uncovered some heterozygous loci in the centromeres. These regions are under active investigation by the T2T consortium to both ensure their structure and understand their evolution[4].

As an additional validation, we investigated potential rare or false collapses as well as rare or false duplications in CHM13v1.0. Here, based on *k*-mer estimates from both GRCh38 and CHM13v1.0 and from Illumina reads for 268 Simon's Genome Diversity Project (SGDP) samples, we identified regions in CHM13v1.0 with a lower or higher copy number than both GRCh38 and 99% of the SGDP samples[212]. We found six regions of rare collapses in CHM13v1.0 that were not in GRCh38 (covering 205 kb, four from one single segmental duplication family). Both our HiFi read depth and Illumina *k*-mer-based copy number estimates suggest these six regions are likely rare copy number variants in CHM13 (e.g.,CHM13v1.0 has only a single copy of the 72 kb tandem duplication in GRCh38, Figure 5.4a). Additionally, we found that CHM13v1.0 had 33x fewer false or rare collapses than GRCh38 (~185 loci covering 6.84 Mbp)[1]. We identified five regions (160 kb) with rare duplications in CHM13v1.0. This included a single 142 kb region that appeared to be a true, rare tandem duplication based on HiFi read depth and Illumina *k*-mer-based copy number estimates (Figure 5.4b). Two of the smaller regions appeared to be true, rare tandem duplications, and two other small regions were identified during polishing as heterozygous or mosaic deletions, revealing potential tandem duplications arising during cell line division or immortalization. In summary, we found 7.5x fewer rare or falsely duplicated bases in CHM13v1.0 relative to the 12 likely falsely-duplicated regions affecting 1.2 Mb and 74 genes in GRCh38[1], including the medically relevant genes: *CBS*, *CRYAA*, and *KCNE1[217]*.

Toward a completely polished sequence of a human genome

While evaluating CHM13v1.0, the T2T consortium successfully completed the construction of the rDNA models and their surrounding sequences on the *p*-arms of the five acrocentric chromosomes[147]. In parallel, we determined that all telomeric sequences remained unpolished. Specifically, in canonical [TTAGGG]n repeats, we found both HiFi read coverage dropouts and ONT strand bias impeded high quality variant calling (Supplementary Figure C.9). For ONT, we observed only negative strands on the *p*-arm and only positive on the *q*-arm across all telomeric repeats at chromosomal ends; we suspect the ONT ultra-long transposon-based library preparation prevents reads from starting at chromosome ends, causing reads to only read into the telomere[136, 89]. We tailored our PEPPER-based polishing approach and performed targeted telomere polishing to remove these errors remaining in telomeric sequences. Finally, automated polishing (described below), indicated that the *FAM156B* gene was heterozygous in CHM13v0.9 and CHM13v1.0 represented the rare minor allele (encoding a premature stop codon) at this locus. We replaced this minor allele with the other CHM13 allele encoding a full-length protein sequence. Overall, we made 454 telomere edits, producing longer stretches of maximum perfect matches to the canonical k-mer at each position across these telomeres compared to CHM13v1.0 (Supplementary Figure C.10). Combined with the parallel completion of the five rDNA arrays, our final round of polishing led to an improved QV of Q73.94 for CHM13v1.1.

Again, to ensure updates did not compromise the high accuracy of the assembly and to identify any remaining issues, we carried out an additional round of SV detection and manual curation using HiFi and ONT with an updated Winnowmap2 alignment (Supplementary Figure C.11), classifying seven loci as remaining issues in CHM13v1.1 (Supplementary Table C.3). We excluded CLR because the lower base accuracy compared to HiFi and ONT and shorter read length compared to ONT were adding no information. Bionano was also excluded as the molecules were lacking coverage in centromeric regions (Supplementary Figure C.12) and did not detect any structural issues beyond the missing telomere and a few heterozygous structural variants already identified by HiFi and ONT. Two loci located in the rDNA sequences appear to be a potential discrepancy between the model consensus sequence and actual reads or an artifact of mapping or sequencing bias. Lower consensus quality is indicated at two other loci, one detected with read alignments that were both low in coverage and identity, and one of which contained error k-mers detected by the hybrid dataset. One locus consisted of multiple insertions (<1kb) with breakpoints detected in low-complexity sequences associated with heterozygous variants and indicated a possible collapsed repeat (Supplementary Figure C.13) and an additional two loci joined and created an artificial chimeric haplotype (Supplementary Figure C.14).

Additionally, we found 218 low coverage loci using HiFi (Supplementary Table C.4), with 81.2% associated with GA-rich (78.0%) regions. The remaining 41 loci had signatures of lower consensus quality and alignment identity, and 30 had error k-mers detected from the hybrid k-mer dataset. In contrast, we detected one low-coverage locus using ONT that overlapped the GA-rich model rDNA sequence. We associated most remaining loci, totalling only 544.8 kb or <0.02% of assembled sequence, with lower consensus quality in regions lacking unique markers. Overall, we found 394 heterozygous regions, including regions with clusters of heterozygous variants (https://github.com/mrvollger/nucfreq), totalling 317 sites (~1.1 Mb).

We manually curated, both the breakpoints and alternate sequences associated with 47 heterozygous SVs, including sites previously inspected (CHM13v1.0) for SV-like error detection. We then investigated HiFi read alignment clippings and confirmed an association with clipping to both true heterozygous variant and spurious low frequency alignments. Additionally, we detected a further heterozygous inversion that went previously undetected.

A comparison to automated assembly polishing

To demonstrate the efficacy of the customized DeepVariant-based approach, we compared our semi-automated polishing approach used to create CHM13v1.0 (Q72.62) to a popular state-of-the-art automated polishing tool, Racon[209, 213]. We iteratively polished CHM13v0.9 (three rounds) using Racon with PacBio HiFi alignments. While the QV improved from Q70.22 to Q70.48 after the first round of Racon polishing, it degraded with the subsequent second (Q70.26) and third (Q70.15) rounds, ultimately diminishing assembly accuracy as a result of overcorrection. We also found that Racon incorporated 7,268 alternate alleles from heterozygous variants identified by DeepVariant, thus potentially causing undesirable haplotype-switching in originally haplotype-consistent blocks. To examine how Racon polished large, highly similar repetitive elements, we counted the number of corrections in non-overlapping 1 Mb windows of the CHM13v0.9 assembly and measured local polishing rates. Unlike CHM13v1.0, Racon polishing showed a clear right-tail in the distribution of polishing rates, indicating the presence of polishing "hotspots", defined here as loci with >60 corrections/Mb (Figure 5.5a). The proximal and distal junctions of the rDNA units (masked from CHM13v1.0 polishing) were prevalent among these loci, a finding that reinforced the importance of masking known collapsed but resolved loci to avoid overcorrection. We also found non-rDNA loci that were preferentially polished by Racon, including satellite repeats such as the highly repetitive HSat3 region in chromosome 9. Finally, CHM13v1.0 made two corrections, recovering two protein-coding transcript's open reading frames (ORFs), but Racon did not make these corrections (Supplementary Table C.2). While CHM13v1.0 did not induce invalid ORFs in any transcripts, Racon made 10 corrections that caused invalid ORFs in 22 transcripts (from nine genes) (Figure 5.5b). Most of these corrections occurred at homopolymer repeats, consistent with our previous findings that homopolymer bias in HiFi reads could lead to false expansion or contraction of homopolymers during polishing.

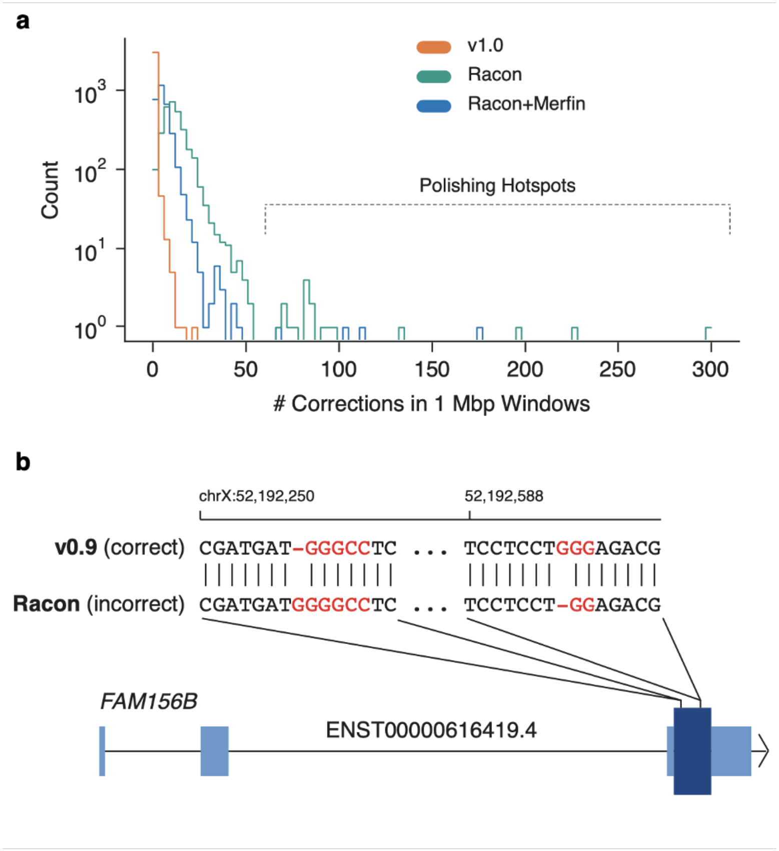


Figure 5.5: Errors made by automated polishing. a, Distribution of the number of polishing edits made in non-overlapping 1 Mb windows of the CHM13v0.9 assembly. b, Two Racon polishing edits causing false frameshift errors in the FAM156B gene. Light blue indicates UTR and dark blue indicates the single coding sequence exon. Highlighted sequence indicates GC-rich homopolymers.

To overcome these relative shortcomings of Racon polishing, we tested polishing the CHM13v0.9 assembly with three iterative rounds of Racon followed by filtering with Merfin (Racon+Merfin). After each round of polishing, Merfin removed proposed Racon edits that incorporated false assembly k-mers. As expected, the Racon+Merfin assembly QV monotonically increased from Q70.22 to Q77.34, Q77.99 and Q78.12. However, Racon+Merfin still incorporated 2,274 alternate alleles from heterozygous variants and polishing hotspots were still evident, suggesting that some repeats were overcorrected (Figure 5.5a). These overcorrections are not reflected in the QV measurements as k-mers from true heterozygous variants are considered 'valid' sequences. Merfin mitigated the 10 ORF-invalidating Racon corrections, however, Merfin also failed to correct the two reading frame corrections made in CHM13v1.0 but not Racon (Supplementary Table C.2). Overall, when considering only automated polishing, we suggest that Racon and Merfin can be used together as a highly effective strategy for building reference assemblies with minimum false positive corrections. However, we would like to emphasize that a custom polishing pipeline with manual interventions is still required for preserving haplotype consistency and avoiding repeat overcorrection.

Discussion

The CHM13v0.9 human genome assembly represented a landmark achievement for the genomics community by representing previously unresolved repeats in a locally haplotype consistent assembly. Though it was imperative to validate and correct this draft assembly, successful polishing faced three major obstacles. First, while repeats are challenging to polish in any draft assembly, the CHM13v0.9 assembly represented hundreds of megabases of exceptionally large and complex repeats genome-wide, which could potentially induce false positive (overcorrection) or false negative polishing corrections. Secondly, though the CHM13 genome is mostly homozygous, we identified non-negligible levels of interspersed heterozygous variation. Therefore, it was essential to distinguish between heterozygous variants and polishing edits in order to maintain the original haplotype consistency. Finally, our evaluation of CHM13v0.9 discovered how homopolymer and coverage bias in HiFi reads caused assembly errors genome-wide. This analysis also revealed that standard methods for measuring QV can be influenced by technology-specific biases.

These obstacles necessitated a custom and contextualised polishing and evaluation model that capitalised on the wealth of available data to exploit the advantages of each sequencing platform. It also required the use of specialized aligners, hard masking, and manual intervention to avoid false polishing corrections within repeats. This polishing approach called for just 1,457 corrections including: *p*-arm of chromosome 18; 454 telomere corrections; 1 large deletion; 2 large insertions; 993 SNPs; 113 small insertions and 880 small deletions. Although the final CHM13v1.1 is highly accurate (Q73.9), we identified 225 loci that were recalcitrant to validation, and we have documented these loci along with 394 heterozygous loci (317 merged loci) (https://github.com/marbl/CHM13-issues/).

The high accuracy of CHM13v1.1 showcases the effectiveness of our informed selection and implementation of appropriate repeat-aware aligners[137, 86], *k*-mer evaluation and filtration tools, and highly accurate and sensitive variant callers[191, 58] whilst also highlighting the utility of capitalising on the synergistic nature of multiple sequencing technology platforms. The minimal number of corrections implemented by our approach and uniform coverage (99.86%) exemplifies the high accuracy of the initial graph construction, with sequencing biases being associated with the remaining coverage fluctuations (223 regions were regions of HiFi dropouts, 77.5% found in GA/TC-rich, and AT-rich satellite sequences such as HSat2/3 and HSat1, were associated with in HiFi coverage increases and ONT coverage depletion, respectively)[147].

In many respects, the T2T CHM13 genome assembly initiative is not representative of typical assembly projects. The success of the CHM13v1.0 assembly was enabled by the low level of heterozygosity of the CHM13 genome, advancements in sequencing technologies, a combination of sequencing technologies (HiFi, ONT, Illumina), customised assembly algorithms, and a large dedicated team of scientists, yielding results currently not possible with limited resources and automated algorithms[150, 27]. However, despite the unique and semi-automated nature of our polishing and evaluation endeavor, recent trends in DNA sequencing and genome assembly algorithms suggest that CHM13v1.1 is just a preview of an imminent wave of high-quality T2T reference genomes in other species[143, 116, 43]. It is therefore critical that the lessons outlined here be incorporated into the next generation of automated bioinformatics tools[86, 137, 209, 58]. For immediate projects, combining data types, using phased reads with repeat-ware alignments, and carefully filtering polishing edits can improve automated polishing accuracy.

Part VI

Fastest clinical diagnosis of a human genome

Chapter 6

Ultra-rapid whole genome nanopore sequencing in a critical care setting

Preamble

This chapter contains the text from the manuscript "Ultrarapid Nanopore Genome Sequencing in a Critical Care Setting"[75] published in New England Journal of Medicine. Professor Euan Ashley led this work at Stanford Medicine in collaboration with the UCSC genomics institute, Google, and NVIDIA. I worked with John E Gorzynski, the lead author of the clinical paper, and Sneha D Goenka, the lead author of the technical article of this project, to derive a pipeline using highly multiplexed nanopore sequencing and variant calling to diagnose critically ill patients. I share the first authorship of the technical manuscript with Sneha D Goenka and John E Gorzynski.

In this work, we developed a nanopore sequencing pipeline that can report pathogenic variants in under 8 hours. We used a high-throughput promethION device for sequencing to run 48 flowcells simultaneously. We paired the sequencing device with a bioinformatics pipeline based on cloud computing. The bioinformatics pipeline consists of basecalling with Guppy, alignment with Minimap2, PEPPER-Margin-DeepVariant for small variant calling, sniffles for structural variant calling, and a variation filtration scheme that surfaces potentially deleterious variants. Finally, we inspect the variants and report back to the primary care physician. We applied this pipeline on 13 patients and successfully diagnosed 6 of them. All diagnoses were validated using a secondary method, and all undiagnosed patients had gone through secondary testing, which also reported no positive findings.

I consider this study a significant milestone in the maturation of Oxford Nanopore. I am humbled to be a part of a study that impacts human life directly. I am thankful for the opportunity and the fantastic experience I had throughout this work.

Introduction

Rapid genetic diagnosis can guide clinical management, reduce cost, and improve prognosis in critically ill patients.[54, 12, 20, 170] Up to 33% of hospitalized children are reported to have an associated genetic disease. The investigations into these diagnoses are associated with 40% longer hospital stays than patients with non genetic disease.[129, 72]

However, the standard of care for return of results from whole genome sequencing is weeks to months.[65]

Several clinical genomics programs have recently responded to the need for faster whole genome genetic diagnosis. These tests range in turnaround time from three to seven days for either a preliminary or full clinical report.[65, 66] The fastest published genome diagnosis was made in 14 hours and 33 minutes, five hours faster than the previous record which had held for more than two years of 19 hours and 10 minutes.[31, 155]

Most clinical sequencing is carried out using “sequencing by synthesis” where short DNA molecules are extended while affixed to a glass slide (Illumina, Inc, San Diego, California).[40, 41, 227, 201] In contrast, nanopore sequencing (Oxford Nanopore Technology, ONT, Oxford, UK) translates changes in electrical conductance to nucleic acid sequence as DNA or RNA molecules transit through a small “nanopore” protein.[179] DNA molecules up to millions of bases in length can be sequenced using this approach. Advantages of sequencing longer molecules include phasing[192, 40, 205] and improved characterization of large genetic variants especially in complex areas of the genome.[133, 127] Historically, barriers to more widespread adoption of long read sequencing in clinical medicine have included cost and per-base error rate, each of which has declined in the last five years.[139, 37]

We sought to take advantage of the improved accuracy and decreased cost of nanopore sequencing to rapidly generate DNA sequence in hundreds of thousands of nanopores across 48 flow cells. We coupled this high sequencing throughput with an accelerated computational pipeline to achieve clinical-grade whole genome sequencing analysis on critical care timescales.

Results

Patient recruitment

Between December 2020 and May 2021, we enrolled a total of 12 patients (five female and seven male, ranging from three months to 57 years, Table 6.1.).

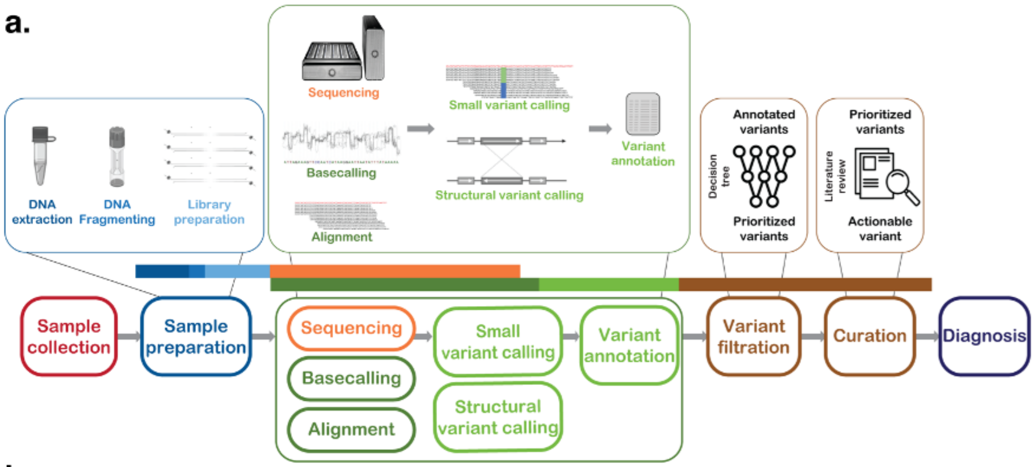
Patient				Variants Calls/Prioritization		
ID	Age	Sex	Ethnicity	SNVS/INDELS	Prioritized 4+	Prioritized SV
1	2 years	M	White	4,419,773	39	22
2	13 years	M	White	4,442,280	28	20
3	6 months	F	Hispanic	4,478,350	35	17
4	5 years	M	Pacific Islander	4,467,180	30	18
5	3 months	M	Hispanic	4,592,381	53	11
6	4 months	F	White	4,500,293	27	37
7	8 months	F	Pacific Islander	4,482,314	37	25
8	6 months	F	Asian	4,503,667	29	20
9	3 months	F	Hispanic	4,619,267	28	35
10	2 weeks	M	White	4,364,225	16	17
11	57 years	M	White	4,315,548	22	16
12	15 years	M	Black	4,770,449	20	22

Table 6.1: Patient demographics and variant call statistics. The prioritization scheme places variants on a scale of 4–12. SV—structural variant; SNV—single nucleotide variant;indel—insertion or deletion.

Primary phase

We continuously improved the pipeline during the initial phase of the program (Figure 6.1, patients 1-7). We began by using a library preparation method that included a barcoding step. Later, we omitted barcoding without impact on variant calling[200].

a.



b.

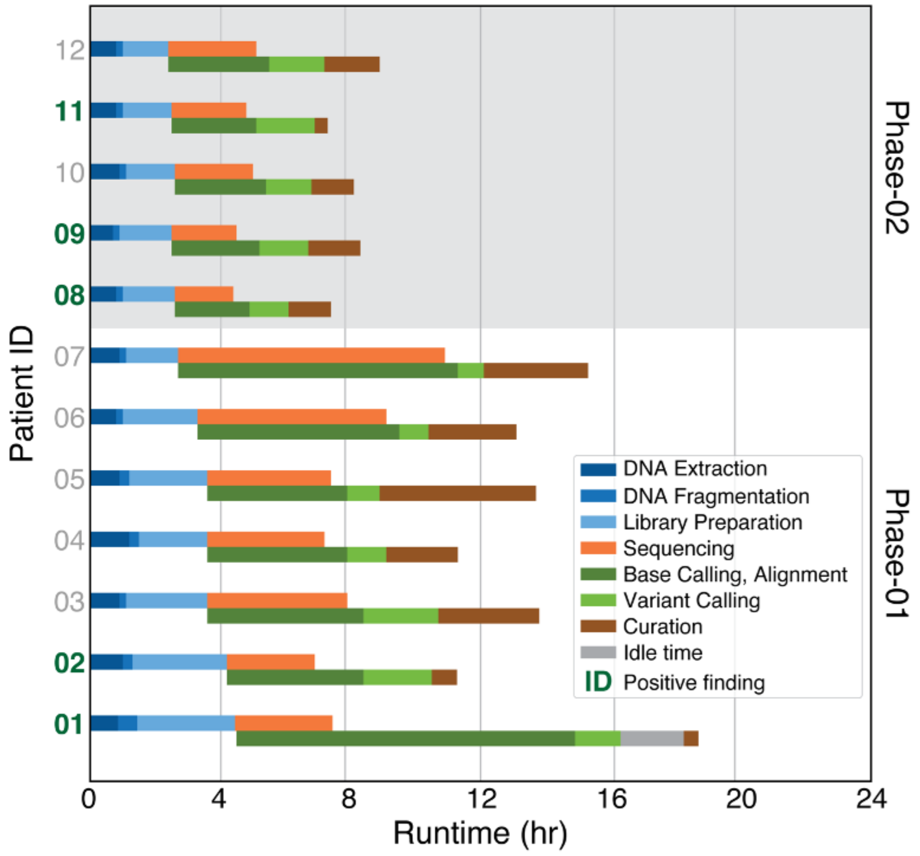


Figure 6.1: (a) The ultra-rapid whole genome sequencing pipeline. The schema depicts all processes from sample collection to a diagnosis. Vertically stacked processes are run in parallel. (b) The performance of the pipeline on twelve patients in two phases. Run-time of individual components are shown by corresponding color from panel (a). The fastest runtime was 7:18 hours (Patient 11) with a positive diagnosis.

This reduced the library preparation time by 37 minutes while increasing input DNA recovery substantially: the mean DNA loading mass per flow cell increased from 155 ng (78 ng—243 ng) to 333 ng (208 ng—345 ng) (Supplementary Figure D.1)). Loading more library per flow cell improved mean pore occupancy from 64% to 82% (Supplementary Figure D.2)).

For patient one, we ran base calling and alignment in a linear fashion which required 7:21 hours of compute time. Starting from patient two, we introduced a pipeline that achieved near real-time base calling and alignment, reducing the runtime beyond sequencing time by 13-fold to a mean of 34 minutes (Figure 6.1).

Secondary phase

For the secondary phase (Figure 6.1, patients 8–12), we introduced two further compute optimizations. The GPU-accelerated version of DeepVariant (NVIDIA Clara Parabricks[151]) reduced the overall variant calling time by 30%. In addition, a local realignment step was introduced to the Deepvariant pipeline which improved variant calling accuracy[200]. We also introduced a new set of flow cells because of pore degradation over the enforced month long winter break precipitated by the COVID pandemic. Together, these changes resulted in a sustained and reproducible increase in the speed and accuracy of the pipeline.

Speed and accuracy

After optimization, the median time taken by each step was shortened (Figure 6.1) such that sample preparation took 2:30 hours (2:24–2:36 hours), sequencing took 2:18 hours (1:48–2.:42 hours), compute took 4:12 hours (3:30–4:48 hours) and curation 1:18 hours (0:24–1:42 hours), reducing the median turnaround time to 8 hours.

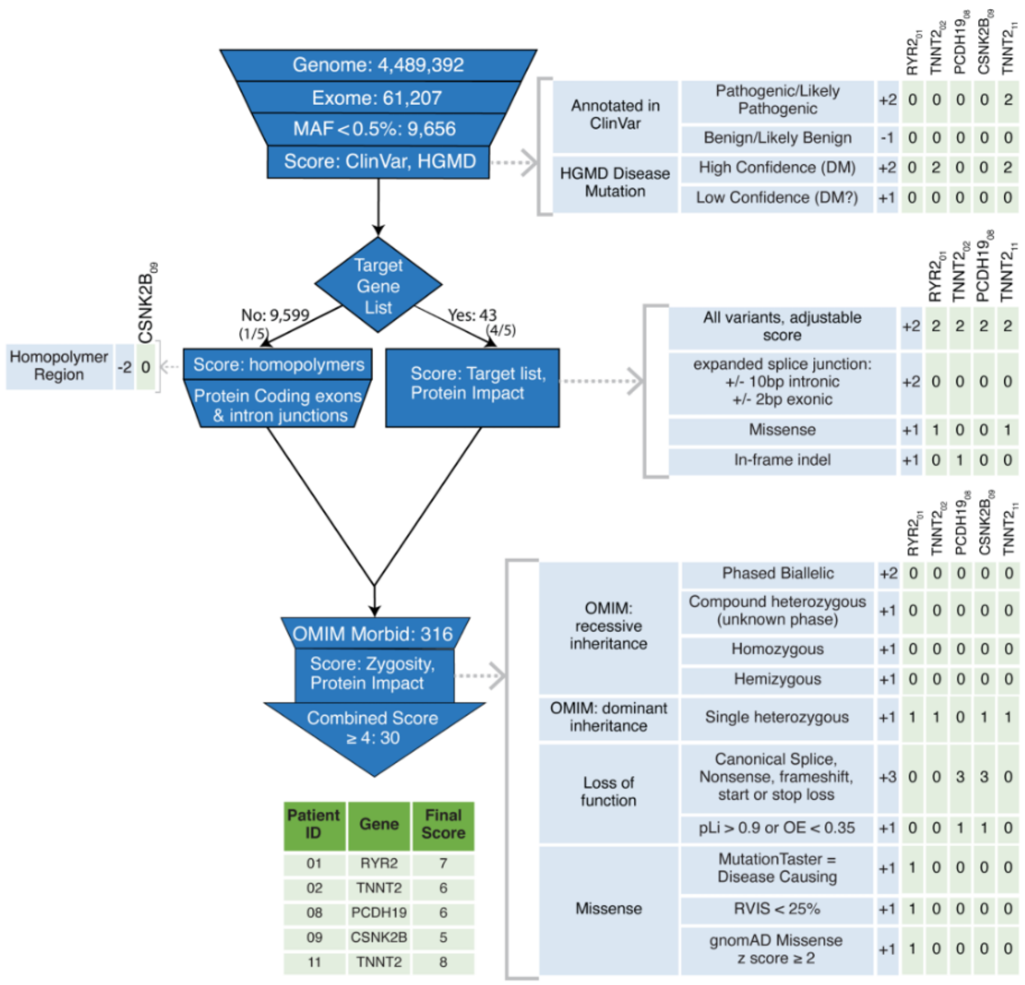
Our fastest recorded total runtime was 7:18 hours (patient 11). The shortest time for each element across the cohort was: DNA extraction—43 minutes (patient 9); DNA fragmentation—10 minutes (patient 10); library preparation—1:25 hours (patient 12); sequencing—1:50 hours (patient 8); overhead base calling and alignment time—21 minutes (patient 11); variant calling—50 minutes (patient 7); and curation—23 minutes (patient 11). Adding these component minimums suggests a theoretical limit for needle to diagnosis with this pipeline of 5:42 hours.

Sequencing, base calling and alignment

We sequenced to a minimum of 170 Gb per genome (173–236 Gb). The average read N50 for these data was 25 kb (20–33 kb, (Supplementary Figure D.4)). We discarded sequence reads that had a quality score of less than seven (assigned "fail" by Guppy). This filtered out 9.8% (4.7–14.1%) of the sequence data. We documented a median alignment identity of 94% (Supplementary Figure D.3)).

Variant Calling and Curation

Across 12 patients, the pipeline surfaced a median of 4,490,490 small variants (single nucleotide variants and small indels). There was a median of 22 structural variants prioritized across 12 cases (Table 6.1.). The mean heterozygous to homozygous and transitions to transversion ratio ratios were 1.5 and 2.0 respectively (Supplementary Figure D.5, Supplementary Figure D.6)))



			$RYR2_{01}$	$TNNT2_{02}$	$PCDH19_{08}$	$CSNK2B_{09}$	$TNNT2_{11}$
Annotated in ClinVar	Pathogenic/Likely Pathogenic	+2	0	0	0	0	2
	Benign/Likely Benign	-1	0	0	0	0	0
HGMD Disease Mutation	High Confidence (DM)	+2	0	2	0	0	2
	Low Confidence (DM?)	+1	0	0	0	0	0

		$CSNK2B_{09}$
Homopolymer Region	-2	0

		$RYR2_{01}$	$TNNT2_{02}$	$PCDH19_{08}$	$TNNT2_{11}$
All variants, adjustable score	+2	2	2	2	2
expanded splice junction: +/- 10bp intronic +/- 2bp exonic	+2	0	0	0	0
Missense	+1	1	0	0	1
In-frame indel	+1	0	1	0	0

			$RYR2_{01}$	$TNNT2_{02}$	$PCDH19_{08}$	$CSNK2B_{09}$	$TNNT2_{11}$
OMIM: recessive inheritance	Phased Biallelic	+2	0	0	0	0	0
	Compound heterozygous (unknown phase)	+1	0	0	0	0	0
	Homozygous	+1	0	0	0	0	0
	Hemizygous	+1	0	0	0	0	0
OMIM: dominant inheritance	Single heterozygous	+1	1	1	0	1	1
Loss of function	Canonical Splice, Nonsense, frameshift, start or stop loss	+3	0	0	3	3	0
	pLi > 0.9 or OE < 0.35	+1	0	0	1	1	0
Missense	MutationTaster = Disease Causing	+1	1	0	0	0	0
	RVIS < 25%	+1	1	0	0	0	0
	gnomAD Missense z score ≥ 2	+1	1	0	0	0	0

Patient ID	Gene	Final Score
01	RYR2	7
02	TNNT2	6
08	PCDH19	6
09	CSNK2B	5
11	TNNT2	8

Figure 6.2: (. Variants are filtered and prioritized through a custom decision tree designed to surface the most likely pathogenic variants. Major filtration steps are depicted in dark blue (numbers represent average number of variants across all samples). Locations of prioritization scoring within the decision tree, as well as possible points assigned for variants meeting each criterion, are listed in light blue; points applied for each reported pathogenic/likely pathogenic variant are listed in adjacent green columns. The final prioritization score of each variant is shown in the dark green table.

Variants from the integrated call file were passed to the filtration tree. An average of 43 (3–93) variants were prioritized in target genes and 309 (167–543) variants outside of target genes. Variants were scored based on several independent factors (Figure 6.2). An average of 219 (111–366) variants received a total score of least one point, with an average of only 14 (7-28) variants receiving a total score of at least five points. All pathogenic or likely pathogenic variants identified in this study scored five or more points. The overall filtration scheme is illustrated in Figure 6.2, with each scoring juncture showing the points available, as well as points scored for each of the five pathogenic variants identified in this study. Of the 22 (11–37) structural variants identified, 16 impacted coding regions.

Diagnosis

We found a likely pathogenic or pathogenic variant in five patients (42%, Table 6.2). These findings were reviewed immediately by study physicians including the care team and a consensus was reached as to whether this variant represented the primary cause of the patient's presentation. All five cases were then rapidly confirmed with an orthogonal technology in a CLIA-CAP laboratory setting. Following those results, clinical action was taken as a result of the findings (Table 6.2). Each finding was judged by a consensus of the treating clinical team to have resulted in actionable findings.

Patient				Disease Associated Variant						
ID	Age	Sex	Presentation	Gene	Location	Details	ACMG Classification	Conf Testing	Diagnosis	Clinical Management Impacted
1	2y	M	Cardiac Arrest, Ventricular Fibrillation	RYR2	c.11621 C>T p.T3874I	Missense, de novo	Pathogenic	MGP	Catecholaminergic Polymorphic Ventricular Tachycardia	Sympathectomy, family screening
2	13y	M	Dilated Cardiomyopathy, NSVT	TNNT2	c.487_489dup p.E163dup	INDEL (STR dup), de novo	Pathogenic	MGP, Sanger	Dilated Cardiomyopathy	Heart Transplant (irreversible cause), family screening
8	6m	F	Very frequent seizures in clusters after 6-month vaccinations	PCDH19	c.2728G>T p.E910*	Predicted truncating, de novo	Likely pathogenic	Sanger	PCDH19-related epilepsy	Selection of anti-seizure medications, access to ongoing clinical trials, counseling regarding prognosis and family planning
9	3m	F	Infantile onset multifocal epilepsy with myoclonic seizures, status epilepticus	CSNK2B	c.73-1G>A	Splice Variant, de novo	Likely pathogenic	MGP, Sanger	CSNK2B-related disorder/ Poirier-Bienvenu neurodevelopmental syndrome	Counseling regarding prognosis and family planning, avoided further extensive work-up
11	57y	M	Left ventricular asymmetrical hypertrophy LV/EF 40%	TNNT2	c. 341C>T p.A104V	Missense, Unknown inheritance	Likely pathogenic	Sanger	Hypertrophic Cardiomyopathy	Cardiac biopsy avoided, heart transplant (irreversible cause), family screening

Table 6.2: Patient description and genetic findings. MGP–Multiple Gene Panel, INDEL–insertion or deletion, NSVT–non sustained ventricular tachycardia, STR–short tandem repeat, LV–left ventricular, EF–ejection fraction, VUS–variant of unknown significance

One case serves as an illustrative example. A 13 year old male previously in good health presented to his primary care provider with a dry cough at night when lying flat, decreased appetite, intermittent chest pain, and fatigue. Thoracic radiographs showed cardiomegaly leading to echocardiography, which revealed a dilated left ventricle with an ejection fraction of 29%. The patient was hospitalized with worsening heart failure and evidence of poor end organ perfusion and shortly thereafter cannulated for veno-arterial extra-corporeal membrane oxygenation (ECMO). The differential diagnosis focused on myocarditis vs genetic cardiomyopathy. Differentiating these became critical to decisions about surgical mechanical support and heart transplant listing--because a significant percentage of acute myocarditis patients recover without the need for advanced therapies, whereas genetic causes in this setting are typically associated with progressive disease. While cardiac biopsy can reveal lymphocytic invasion, it is an invasive procedure with limited negative predictive value because of the patchy nature of the infiltration. We enrolled the patient in our study, and within 11:30 hours of the samples arrival in the lab, we identified a heterozygous duplication within a short tandem repeat element in troponin T (*TNNT2)*, an integral component of the cardiac sarcomere and a gene known to be associated with dilated cardiomyopathy. CLIA-based Sanger sequencing confirmed the presence of this variant and parental testing later revealed it to be *de novo*, confirming this variant as likely pathogenic. While a preliminary result from cardiac biopsy was available nine hours after the biopsy was performed (showing myocyte hypertrophy and patchy interstitial fibrosis with no signs of lymphocytic infiltration), the final pathology report including further staining, immunohistochemistry, and electron microscopy was not returned until ten days later. Thus, our pipeline identified the pathogenic variant less than twelve hours after blood draw, and confirmatory CLIA-based Sanger testing was available prior to the discussion of transplant listing. In contrast, a clinical panel sent to a commercial laboratory did not return results, with the *TNNT2* variant, until after the transplant listing decision, emphasizing the impact of rapid turnaround testing. The patient received a heart transplant 21 days after listing.

Discussion

Genetic disease is disproportionately responsible for extended critical care hospitalization, especially in younger patients. Here, we describe a nanopore-based clinical whole genome sequencing pipeline that can return actionable genetic diagnoses in under eight hours.

Standard time to return of results for whole genome sequencing is 8–12 weeks with rapid protocols returning finalized results in seven days[65] or preliminary results in three days.[66] The fastest genome diagnosis to date from the start of sample preparation to final diagnosis was completed in 14:33 hours.[155] To achieve a 50% drop in the fastest time reported to a genetic diagnosis (7:18 hours) with an average time of 7:58 hours, we optimized every component of the sequencing and analysis workflow for speed and accuracy. This increase in speed did not come at a cost of diagnostic rate (42%) or actionability (all diagnoses were judged actionable by the clinical study and bedside care team). Indeed, long read sequencing at this depth harbors theoretical advantages for the diagnosis of those presenting acutely with severe genetic disease.

Nanopore sequencing lends itself to parallelization. Here, we demonstrate that nanopore flow cells can be deployed in an interval manner to sequence multiple genomes to high depth. While 48 flow cells can generate in one hour data equal in size to that typically generated for a short read clinical genome (40x coverage, S1 flow cell, 150 bp reads, 100 gigabases) in around 16 hours[70, 31] prior modeling of nanopore data[192] led us to aim for 50-60x coverage (minimum 170 gigabases, with an N50 of 20-25kb) since that had been shown to equal short read consensus accuracy. Converting base calling and alignment to real time at this scale has not previously been achieved. The cloud solution involved optimization of transfer, storage, and input-output of data to the dynamically tasked central or graphical processing units. In fact, because of the parallel nature of this solution, it would be possible not only to meet the theoretical limit of just over five hours for one genome, but by overlapping processes to fully complete sequencing and analysis of three genomes within a total time of nine hours.

Many groups have demonstrated the potential of exome or genome sequencing to solve undiagnosed disease in outpatient settings.[201, 227, 226, 105] Others have demonstrated the time and cost savings of rapid diagnosis in neonatal or pediatric critical care settings.[20, 31, 54, 170, 171, 203] Cost savings are predicated on shortening the significant expense of even short stays in a critical care setting (often over \$10,000 per day). Securing a confident genetic diagnosis within one to two days was shown to lead to improved prognosis and dramatically reduced costs in a neonatal population.[54] Here, we accepted patients of all ages and surfaced results within the time scale of a single nursing shift. Our patients were broadly similar to those previously reported in rapid genome sequencing studies: most patients harbored primary neurological or cardiovascular presenting signs. Such presentations are associated with genetic diagnostic rates of 30-50% and high levels of actionability. These data represent some of the earliest evidence that rapid turnaround genome sequencing can offer similar benefits to adults as has been clearly shown for neonates.

In addition to speed, long read sequencing harbors advantages for the diagnosis of critical illness of genetic etiology including the ability to simultaneously assess DNA sequence and methylation state. We previously demonstrated the benefit of long read sequencing in the diagnosis of a 2.2kb structural variant undetected with short read approaches.[133] While algorithms can be tuned to diagnose structural variation and genomic repeats from short read (150bp) sequencing data,[22, 42] long read sequencing with median read lengths upwards of 15 kb offers advantages for the characterization of genome variations spanning distances up to tens of thousands of base pairs.

In summary, we develop, test, refine, and implement a novel approach to ultra-rapid clinical whole genome sequencing diagnostics across adult and pediatric critical care environments. We show that for comparable cost to existing approaches, our method can achieve turnaround times from needle to diagnosis of less than eight hours, returning actionable, cost-saving diagnostic information within the course of a single hospital shift.

Methods

Patient recruitment

Enrollment was open to any critical care patient at Stanford hospitals (Stanford Health Care and Lucile Packard Children's Hospital) with a clinical presentation consistent with a genetic disease. Priority was given to patients where a rapidly identified genetic diagnosis would be clinically impactful for the patient or the patient's family. We acquired consent from adults directly and for minors, from parents or guardians according to Stanford IRB protocol 58559.

Sample collection and preparation

We collected an average of 2ml of whole blood. The preparation of a sequencing library for distribution over 48 flow cells requires a substantial yield of high quality, high molecular weight genomic DNA. We optimized DNA extraction and library preparation for yield, quality, and speed using a modified Puregene (Qiagen, Hilden, Germany) genomic DNA extraction protocol, from the limited volume of blood. DNA was fragmented using g-TUBE (Covaris, Massachusetts, USA) and sequencing libraries were prepared with an SQK-LSK109 (Oxford Nanopore, Oxford, UK) protocol modified such that eight libraries were sufficient to load 48 PromethION flow cells. For barcoded samples, an additional EXP-NBD104 kit was used for barcoding. A detailed description of these methods can be found at DOI 10.17504.

While one PromethION flow cell can generate 200 gigabases (Gb) of data in 2-3 days of sequencing, our goal was to complete sequencing as quickly as possible. To this end, we distributed a DNA library from a single patient equally over 48 PromethION flow cells (Oxford Nanopore, Oxford, UK) and sequenced until we achieved a target range of greater than 170 Gb of data. After method optimization, we were able to generate as much as 200 Gb of data in as little as 1 hour 50 minutes. We dramatically reduced the sequencing cost-per-sample by washing the flow cells after sequencing and reusing the flow cells for multiple samples.

Base Calling and Alignment

Sequencing from 48 flow cells can yield more than 100 Gb per hour. While the PromethION tower is capable of locally base calling and aligning the data generated during sequencing, it cannot keep up with such a high rate of data generation. This increases the overhead compute time — base calling and alignment runtime beyond completion of sequencing — to almost 18 hours (220 Gb output).[200] To mitigate this, we implemented an approach that scaled multiple PromethION-tower like compute instances in a cloud computing environment (Google Cloud Platform) to achieve near real-time base calling and alignment of sequencing data. Going from a local to a cloud-based pipeline, we developed a framework where terabytes of data were transferred to the cloud storage in real-time during sequencing and distributed across the instances so as to minimize the tail latency. Specifically, as soon as sequencing begins, a script starts to periodically upload raw signal output files from the local tower to the cloud storage. Simultaneously, multiple compute instances fetch batches of signal files designated to the particular instance from the cloud storage to perform base calling (Guppy v4.2.2) and alignment (Minimap2 v2.17-r974[111]) concurrently. The reads were aligned to the GRCh37 human genome reference genome.

Variant Calling

After alignment, we used a haplotype-aware long-read optimized variant caller (PEPPER-Margin-DeepVariant[192]) to identify single nucleotide variants (SNVs) and small insertions and deletions (indels). The pipeline included the structural variant caller Sniffles.[189] The pipeline is parallelized over multiple compute instances, each processing a specific set of contigs in order to rapidly generate an integrated variant call file.

Annotation

Small variants in homopolymer regions were annotated for use in the prioritization scheme. Structural variants were annotated with the frequency of similar variants in public and in-house SV catalogs derived from both short-read and long-read sequencing studies. Rare SVs that overlapped coding sequences were ranked based on the impacted gene and non-coding rare SVs were placed in a second tier of variants for curation.

Variant filtering and prioritization

Patient-specific gene target lists were developed for each case, in collaboration with ordering clinicians. While variant prioritization was not limited to the gene list, it ensured that genes highest on the clinical differential were examined rigorously. Alissa Interpret software (Agilent, Palo Alto, CA) was used to filter and prioritize likely deleterious variants, using a custom classification tree. Analysis was limited to variants that had gnomAD minor allele frequency<0.5% (in all non-bottleneck subpopulations) and that were in genes either associated with disease in the Online Mendelian Inheritance in Man catalog ("OMIM morbid") or included on the target list. Variants not on the target gene list were further limited to those either within protein coding exons (including +/-2 intronic bases), or those with high confidence annotations in both the Human Gene Mutation Database (HGMD) and ClinVar.

Variants were then prioritized for review based on: (i) annotation in ClinVar and/or HGMD, (ii) inclusion on the target gene list, (iii) patient zygosity consistent with OMIM-annotated inheritance for associated disease, and (iv) predicted deleteriousness (e.g., truncating or missense variants in a gene highly intolerant of missense variation) (Figure 6.2). The prioritization scheme allowed rapid review and interpretation of likely actionable variants by setting a threshold of 4 or higher for review, and elevating the highest scoring variants for focused interpretation.

Variant curation and molecular board review

Variant interpretation was performed by one genetic counselor (MEG), one genomic scientist (DGF) and one American Board of Medical Genetics and Genomics board certified molecular geneticist (ES). This team worked in parallel, dynamically dividing and discussing variants as needed, and reviewed variants in order from the highest score (max=12) to the lowest (min=4).

If a likely pathogenic, pathogenic, or otherwise suspicious variant was reported, it was reviewed in detail on a conference call by the clinical genetics and care team. If a consensus was reached that a pathogenic or likely pathogenic variant consistent with the patient's phenotype was found, clinical confirmatory testing was completed either via single-site Sanger sequencing in Stanford's CLIA-CAP Clinical Genomics lab, and/or via clinical panel sequencing at an external CLIA/CAP laboratory.

Part VII

Discussion

Chapter 7

Discussion

Genomic studies enabled by next-generation short-read sequencing technology can identify small variants in mappable regions of the region [232]. Many studies that use a incomplete linear reference or short-reads, skip repetitive regions as they are either missing from the reference or short reads cannot resolve them [30]. The reference materials built using short-read sequencing limit our scientific understanding to only the mappable regions of the genome [30, 153, 221, 135, 120].

Long-read sequencing technology emerged to solve these issues [194, 193, 153]. We have seen the telomere-to-telomere (T2T) consortium generate the first complete human genome and unraveling the biology of centromere and telomere for the first time using long reads [135, 148]. The Human Pangenome Reference Consortium aims to create genome assemblies across populations to improve variation inference across the human population [78]. However, much of the improvements with long-reads are hampered by scalability issues and error-rate of the platform.

Oxford Nanopore sequencing platform offers a portable and affordable solution to genome sequencing but suffers from a unique error mode [174, 119]. The presented dissertation demonstrates the ability to use deep learning to improve the genome inference quality of nanopore sequencing. In chapter 3, I presented a haplotype-aware variant calling pipeline for nanopore that outperforms short-reads in single-nucleotide variants at whole genome-scale. In chapter 4, I presented a *de novo* assembly pipeline that can scale to a population scale. In the final two chapters, chapter 5 and 6, I presented the applications of the methods we developed to validate the first complete human genome assembly and applying nanopore sequencing in a clinical setup.

Nanopore sequencing has shown some significant improvements over the last few years [119, 174, 225]. The quality of reads for DNA sequencing has improved from 60% in 2014 to 98.4% in 2022 [174, 190]. As a result, we have seen improvements in genome inference quality [193, 190, 153]. We have also seen advances in nanopore chemistry that translate into downstream high-quality genome inference [190]. With many more nanopore sequencing applications like direct RNA sequencing [211], protein sequencing [16], and targeted sequencing [164], the genomics community will significantly benefit from the maturation of this platform.

One of the major issues with nanopore genome inference is small INDEL accuracy

[194, 193]. The highest INDEL F1-score observed with nanopore is 0.88 with R10.4 Q20 data, whereas Illumina and PacBio HiFi routinely report a 0.995 INDEL F1-score [166, 221]. The non-randomness of INDEL-like errors in repeat regions makes it difficult to identify small INDEL variants correctly [174, 193]. In many cases, the errors are indistinguishable from true variants. Even though nanopore provides high SV calling accuracy [38], methylation accuracy [196, 68] and generates contiguous genome assemblies [119, 194], the small INDEL issue is currently the one we need to address. I believe providing the deep learning models with the information of the source of these errors, the nanopore signal, would improve inference. However, encoding nanopore signals in the inference process come with several roadblocks.

Encoding the nanopore signal information to the deep learning models needs improvement on many fronts. First, deep learning-based nanopore basecallers are proprietary, so it is not possible to customize output information quickly integrated like likelihood of each sate. A secondary option is to align the signal information back to the read sequences to derive a set of signal features. Methods like Nanopolish [196], SignalAlign [173] did this routinely. However, these methods often encounter the issue of long running time. The nanopore devices output signals in HDF format that requires a significant amount of storage and HDF format does not support parallelization. Not having the ability to parallelize and the required storage limits the usage of nanopore signals in downstream applications. Some recent work in nanopore signal storage has shown improvements but requires further improvement to have this integrated for routine method development[62].

Nanopore sequencing is entering a very exciting time. With a vibrant community that supports the development of this sequencing platform, it is unlikely that we will not see breakthroughs in the coming years. With the human pangenome effort, we will have some new challenges in mapping nanopore reads to pangenome graphs and identifying variants correctly [78]. The telomere-to-telomere consortium published the first complete human genome [148]. Routine large-scale studies of genomic variation in the most challenging regions[2] in the T2T genome will depend on how much improvement we will see in the sequencing platforms that can generate long reads, including nanopore.

It has been a humbling experience to participate in some groundbreaking work that nanopore has enabled. I firmly believe the community will move this forward. Long-read genome sequencing has been invaluable to improving the scientific understanding of our genomes. I hope it will soon show the path to making the genomic analysis as routine as regular blood testing.

Part VIII

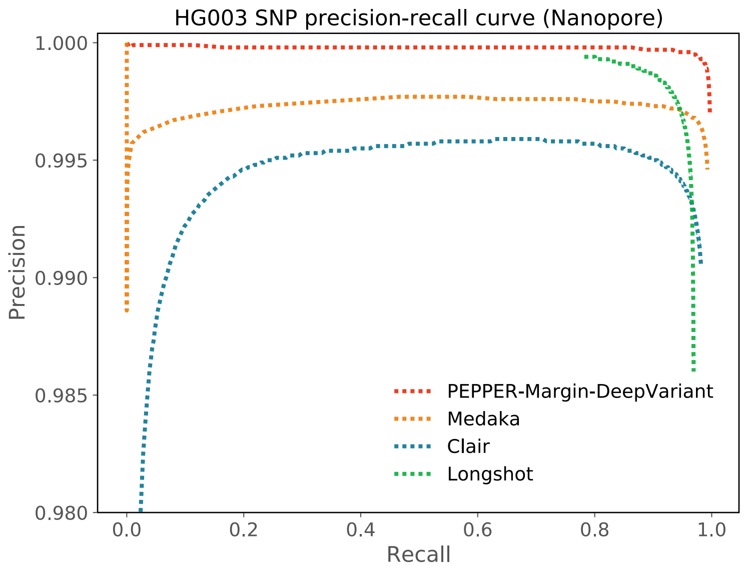
Appendices

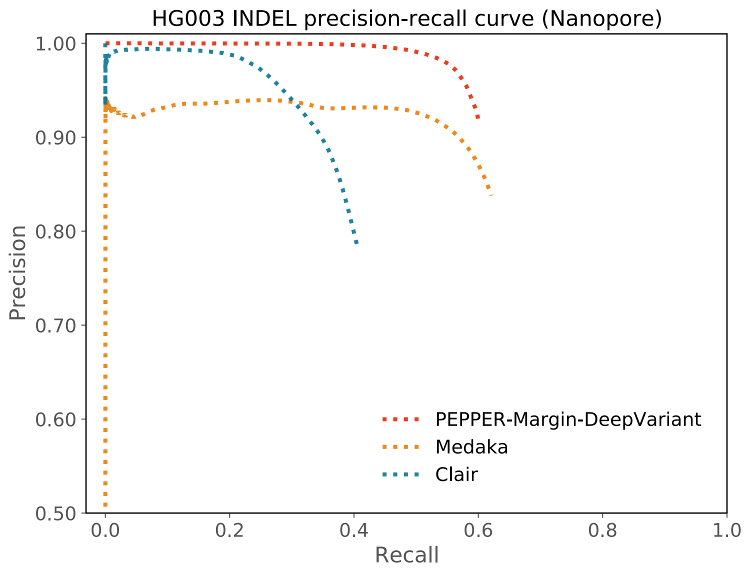
Appendix A

Appendix A: Supplementary information for haplotype-aware variant calling with PEPPER-Margin-DeepVariant

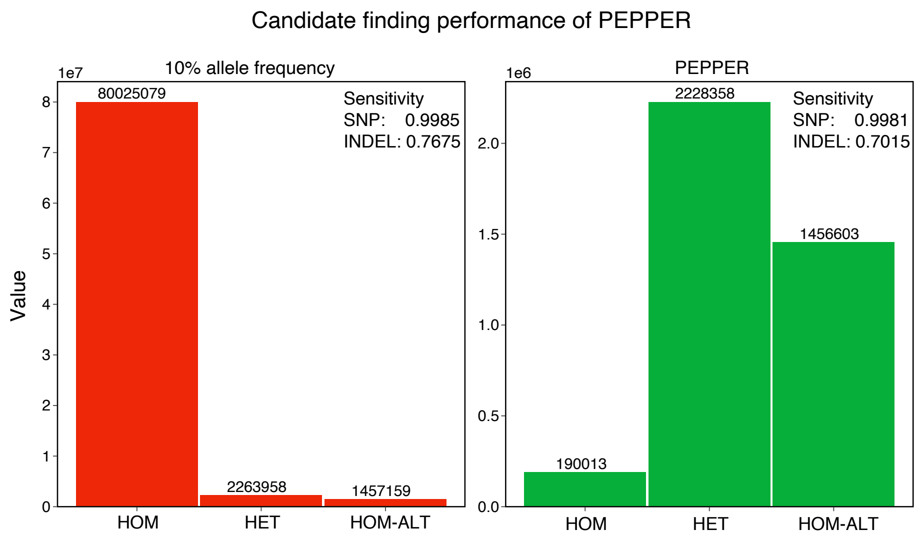
Preamble

Supplementary Figures

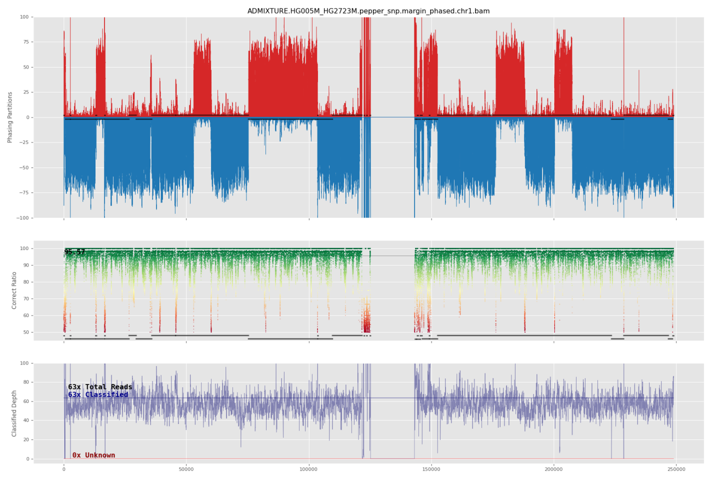




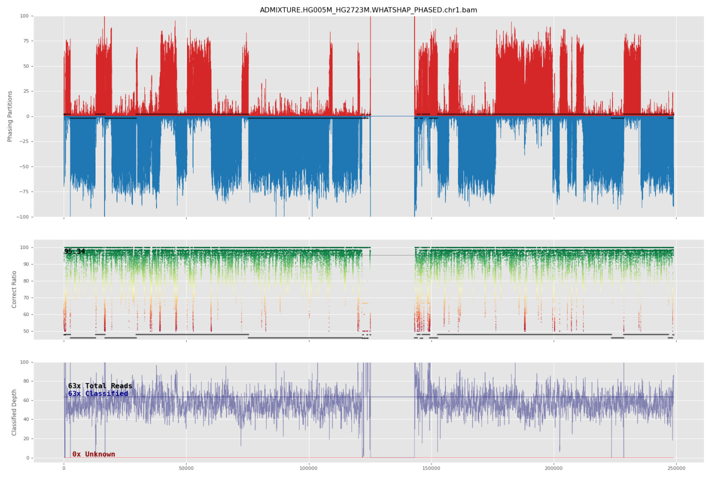
Supplementary Figure A.1: Precision-Recall plot of HG003 for nanopore-based variant callers.



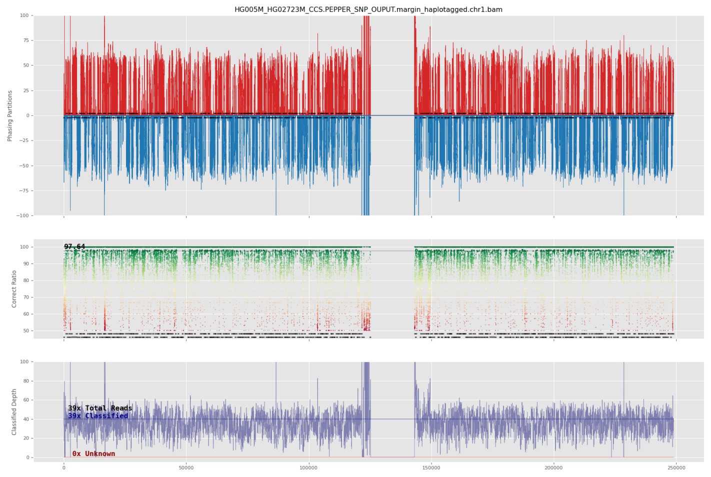
Supplementary Figure A.2: HG003 ONT 90x candidate finding performance comparison between 10% heuristic based approach and PEPPER.



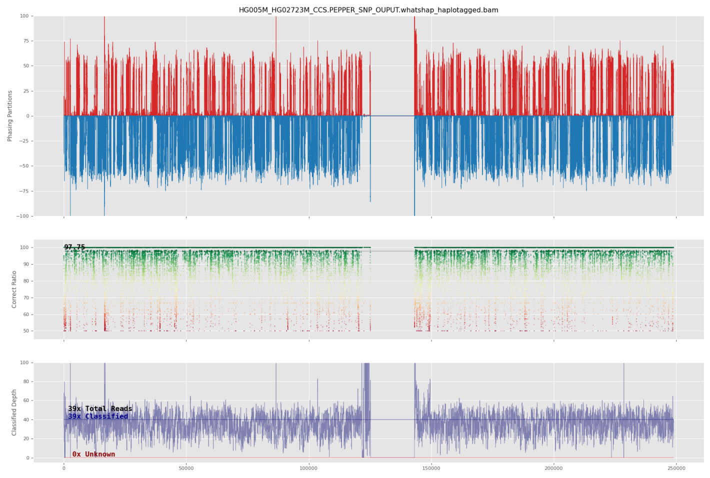
Supplementary Figure A.3: Full Natural Switch plot for chr1 of an admixture of HG005 and HG02723's maternal haplotypes from nanopore data phased by Margin



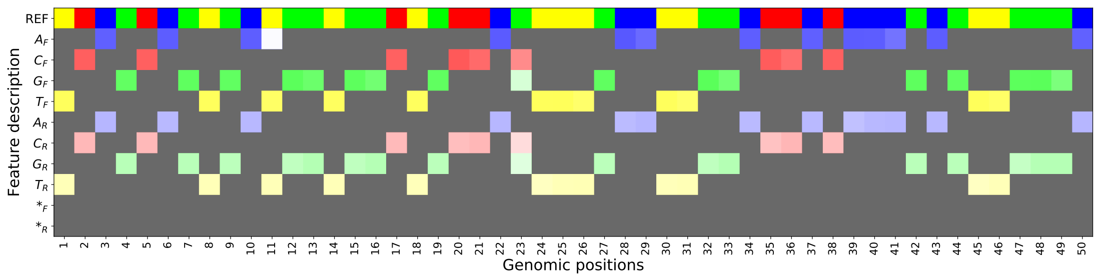
Supplementary Figure A.4: Full Natural Switch plot for chr1 of an admixture of HG005 and HG02723's maternal haplotypes from nanopore data phased by WhatsHap



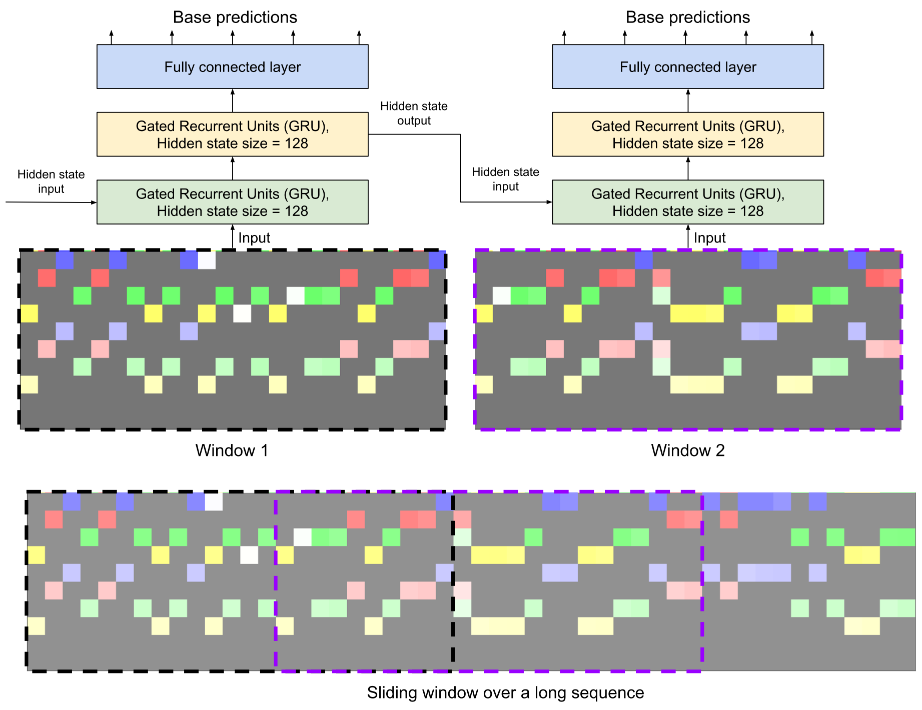
Supplementary Figure A.5: Full Natural Switch plot for chr1 of an admixture of HG005 and HG02723's maternal haplotypes from PacBio HiFi data phased by Margin



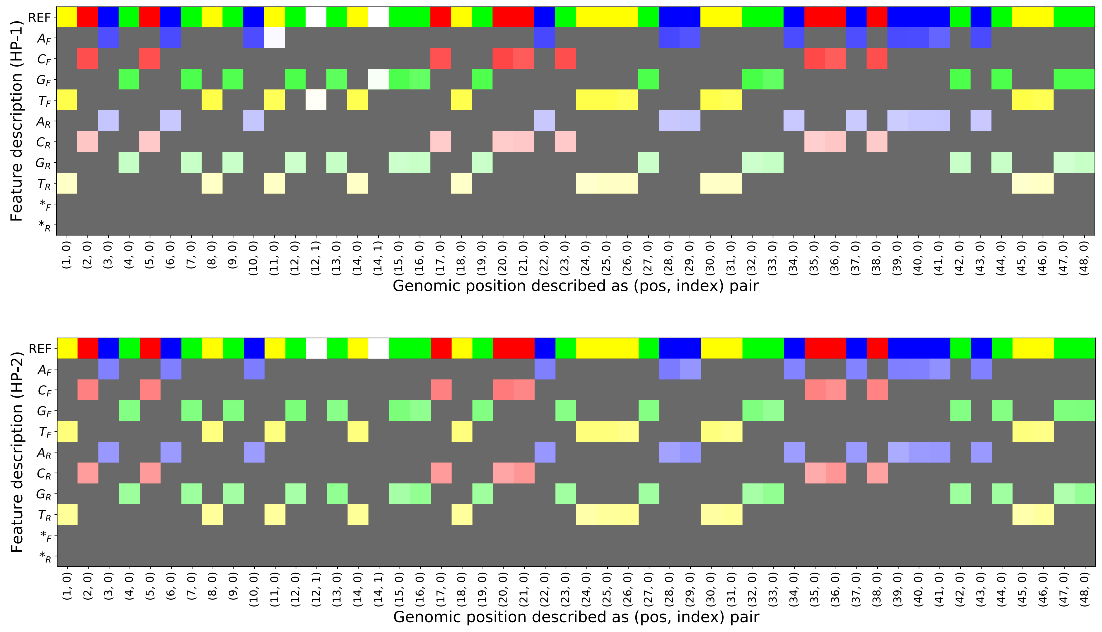
Supplementary Figure A.6: Full Natural Switch plot for chr1 of an admixture of HG005 and HG02723's maternal haplotypes from PacBio HiFi data phased by WhatsHap



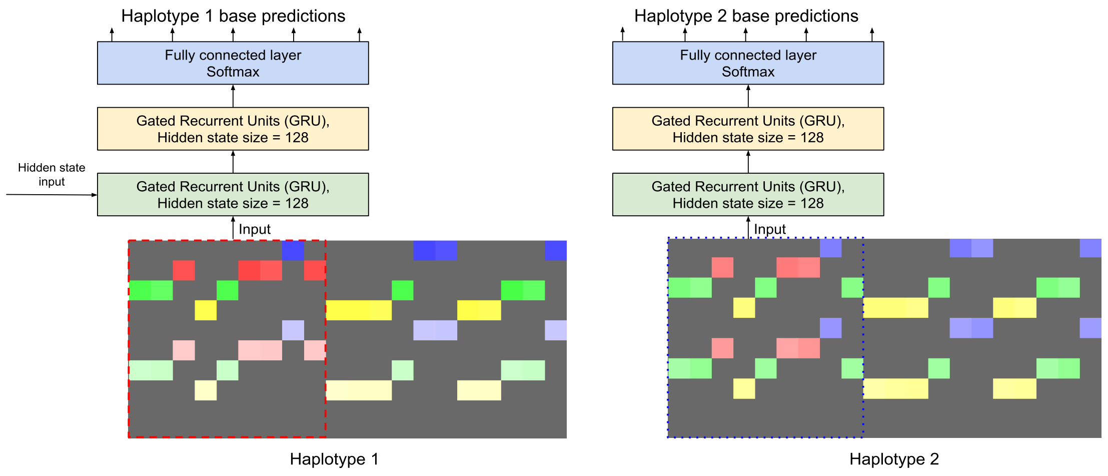
Supplementary Figure A.7: PEPPER-SNP image generation scheme.


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 A.8: PEPPER-SNP inference scheme.



Supplementary Figure A.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).



Haplotype 1 base predictions
Fully connected layer
Softmax
Gated Recurrent Units (GRU), Hidden state size = 128
Gated Recurrent Units (GRU), Hidden state size = 128
Hidden state input
Input
Haplotype 1
Haplotype 2 base predictions
Fully connected layer
Softmax
Gated Recurrent Units (GRU), Hidden state size = 128
Gated Recurrent Units (GRU), Hidden state size = 128
Input
Haplotype 2

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

Supplementary 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 A.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.

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 A.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.

Sample	Ref. GRC	GIAB version	Variant Type	Method	True pos.	False neg.	False pos.	Recall	Precision	F1-score
HG005	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
			INDEL	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
			INDEL	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 A.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).

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 A.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 A.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 (`https://cloud.google.com/products/calculator`). Logs of all the runs are publicly available (See supplementary Notes).

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 A.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 A.14: SNP Variant accuracy statistics for HG003 and HG004 against GIAB v4.2.1 on GRCh38 using PacBio CLR 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)	HG003	SNP	3161145	Illumina	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	Illumina	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
	HG004	SNP	3180117	Illumina	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	Illumina	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)	HG003	SNP	166352	Illumina	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	Illumina	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
	HG004	SNP	166493	Illumina	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	Illumina	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 A.19: Performance comparson of Illumina, PacBio HiFi and Oxford nanopore data in repeat and non-repeat regions.

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 A.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 A.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 A.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 Memory	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 A.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 A.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 A.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 A.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 A.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	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 A.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 A.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.

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 A.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 A.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 A.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 A.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 A.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 A.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 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 A.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	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 A.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 A.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 A.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)	HG003	SNP	2717833	Illumina	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
		INDEL	155063	Illumina	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
	HG004	SNP	2732800	Illumina	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
		INDEL	156444	Illumina	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 A.18: Performance in not all difficult regions (easy regions) with Illumina, PacBio HiFi and Oxford nanopore data.

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 A.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.

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 A.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.

Appendix B

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

Preamble

Supplementary Results

Supplementary Table B.1: Read N50s stratified by sample and flowcell 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 B.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 B.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 B.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 B.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

Supplementary Table B.5: QUAST assembly metrics of three samples on four assemblers.

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
	# 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
	# 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
	# 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

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

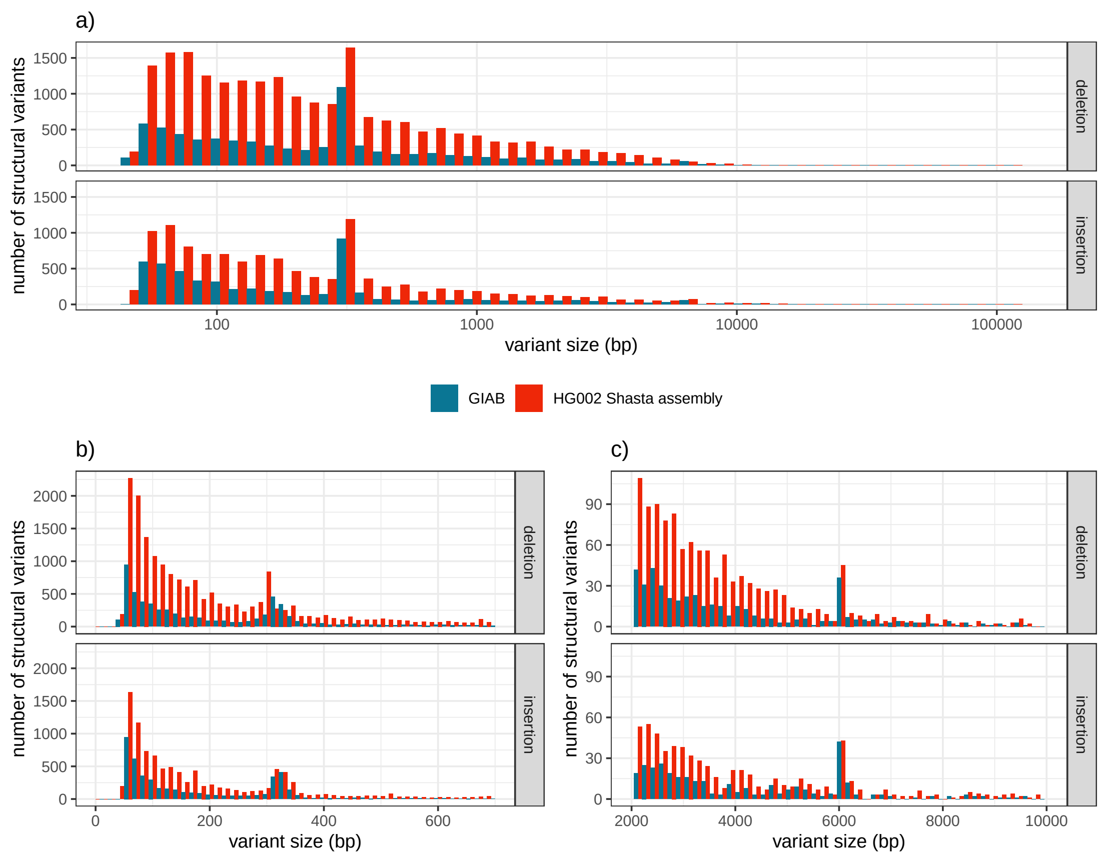
Method	Sample	Minutes	Threads Used	Peak Memory	AWS Instance Type	AWS Instance Cost
WTDBG2	HG00733	2971	63	365	r5a.16xlarge	$3.62
	GM24385	1752	63	293	r5a.16xlarge	$3.62
	CHM13	1655	63	312	r5a.16xlarge	$3.62
WTDBG2 (wtpoa-cns)	HG00733	248	31	12	r5a.16xlarge	$3.62
	GM24385	274	24	12	r5a.16xlarge	$3.62
	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 B.15: Runtime breakdown for each step of the Shasta assembler.

Sample	Input	MinHash	Alignments	Marker graph creation	Transitive reduction	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 B.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 B.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 B.17: CHM13 MHC unpolished Shasta assembly as compared to the nearest matching haplotype in hg38 (GL000251.2)

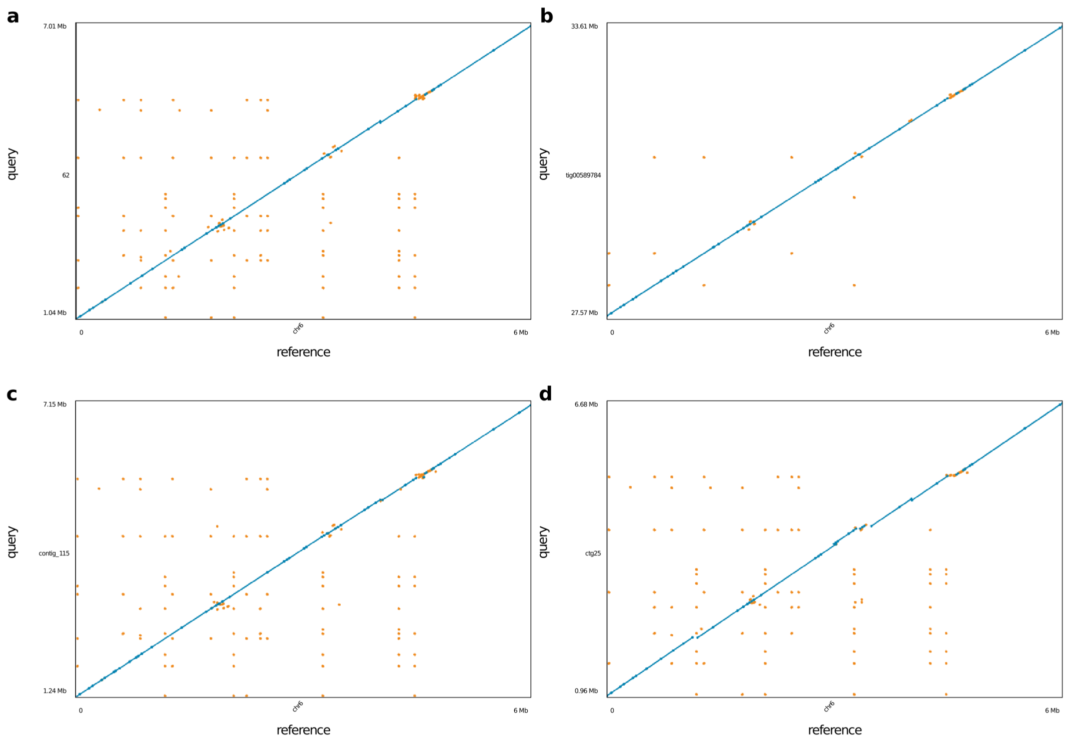
Assembler	Best Contig	Disagreements	Largest Aligned	Mismatch Rate	Indel Rate
Shasta	62	6	2,788,362	0.00296	0.00399
Canu	tig00589784	5	2,792,139	0.00331	0.00607
Flye	contig_115	6	2,787,570	0.00543	0.01106
wtdbg2	ctg25	32	1,819,753	0.00553	0.00576

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

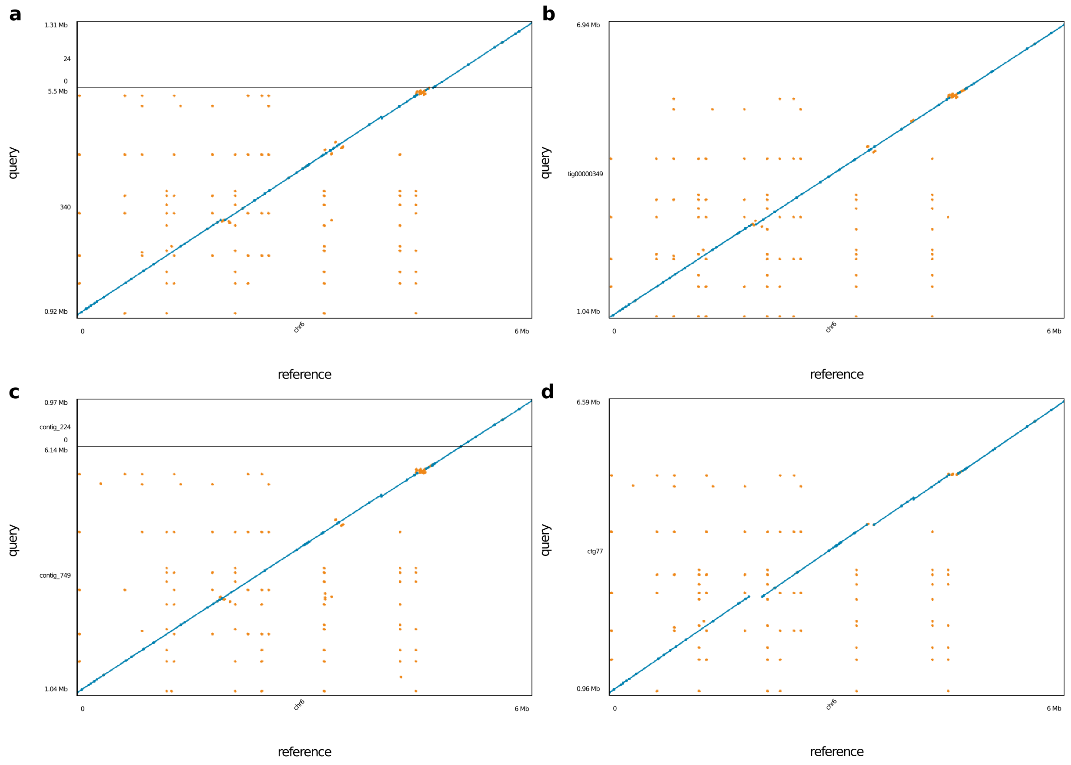
Metric	HG00733-Mother			
	Shasta	Wtdbg2	Flye (initial)	Canu
# contigs	1,934	4,028	1,634	877
Total length	2,754,225,214	2,690,619,717	2,791,893,188	2,829,920,708
N50	9,071,623	14,125,235	25,658,831	19,451,828
NG50	7,702,138	10,217,387	23,775,989	16,507,795
# disagreements	705	3,661	6,082	2,161
Genome 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 B.19: HG00733 Maternal trio binned MHC unpolished Shasta assembly as compared to the nearest matching haplotype in hg38 (GL000255.1)

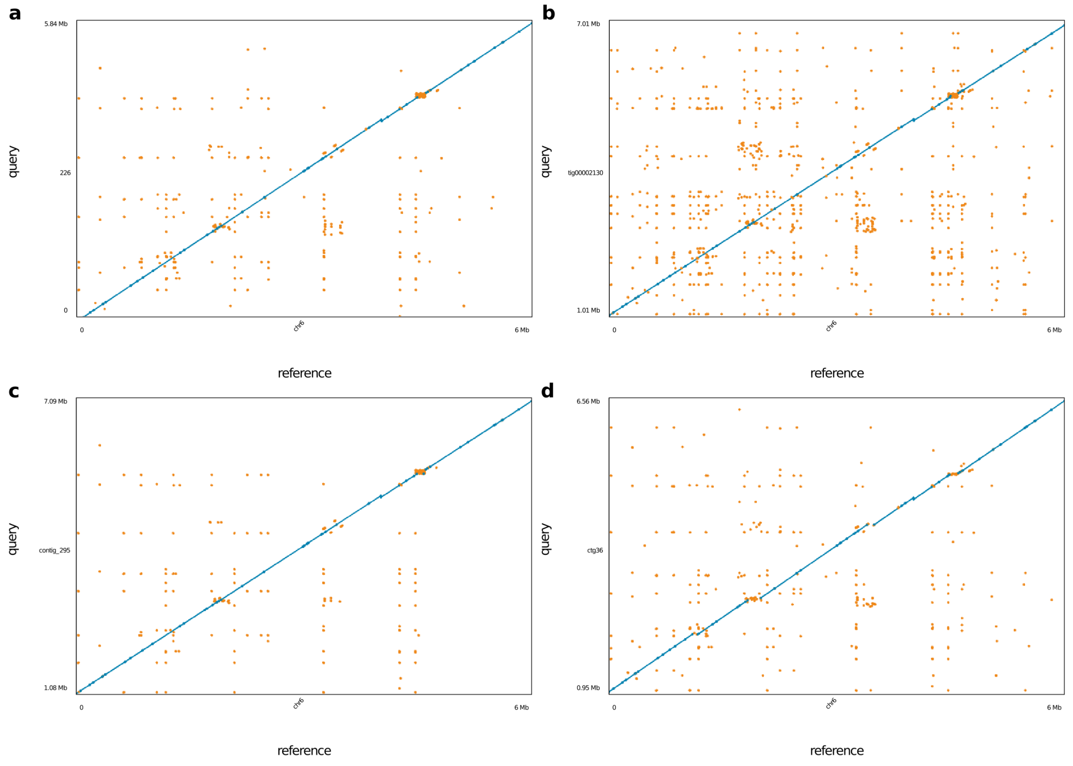
Assembler	Best Contig	Disagreements	Largest Aligned	Mismatch Rate	Indel Rate
Shasta	226	0	4,289,729	0.00206	0.00538
Canu	tig00002130	0	4,289,729	0.00182	0.00676
Flye	contig_295	0	4,289,729	0.00579	0.01759
wtdbg2	ctg36	23	1,418,939	0.00592	0.00905



Supplementary Figure B.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 B.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 B.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.

Supplementary Table B.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 B.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 B.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 Table B.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 B.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.

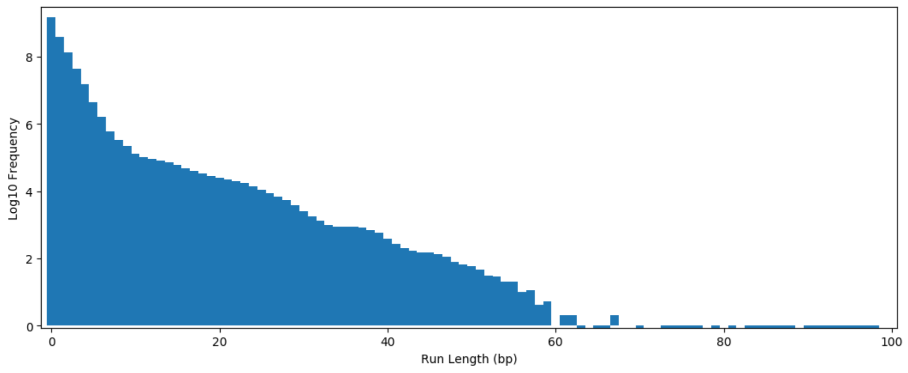
Sample	Assembly	Percentage Errors				Q Scores			
		Balanced	Identity	Deletion	Insertion	Balanced	Identity	Deletion	Insertion
CHM13 ChrX	Shasta (polished)	0.064%	0.006%	0.036%	0.022%	31.92	42.40	34.42	36.51
	Pilon 1x	0.025%	0.004%	0.012%	0.008%	36.06	43.75	39.16	40.75
	Pilon 2x	0.023%	0.004%	0.012%	0.007%	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 B.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 B.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	



Supplementary Figure B.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 B.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 B.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 B.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 B.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 B.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 B.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%

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

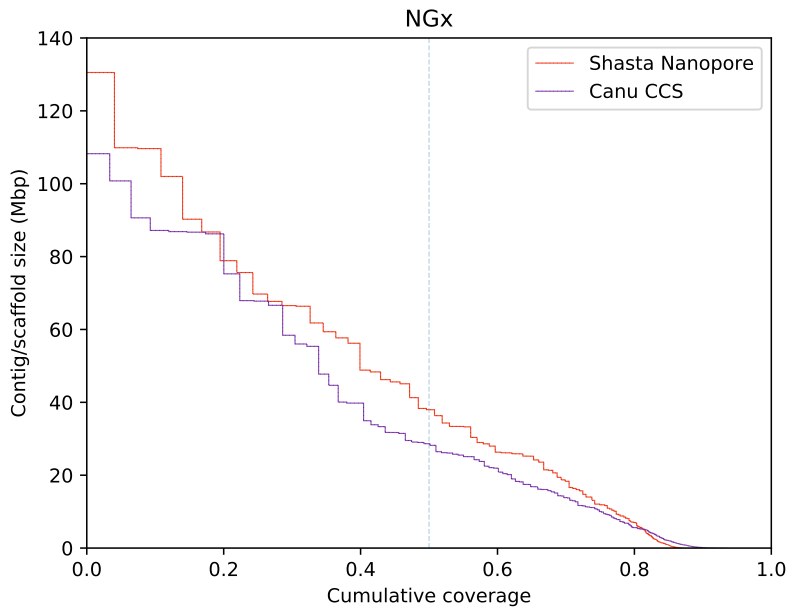
Metric	CHM13	
	Nanopore Shasta MarginPolish, HELEN	PacBio-HiFi Canu Racon
# contigs	1622	5206
Total length	2819245173	3031026325
N50	46206794	29522819
NG50	41255275	29092230
# disagreements	1107	8666
# disagreements outside Centromeres	801	2999
# disagreements outside centromeres and Seg Dups	314	893
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 B.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.

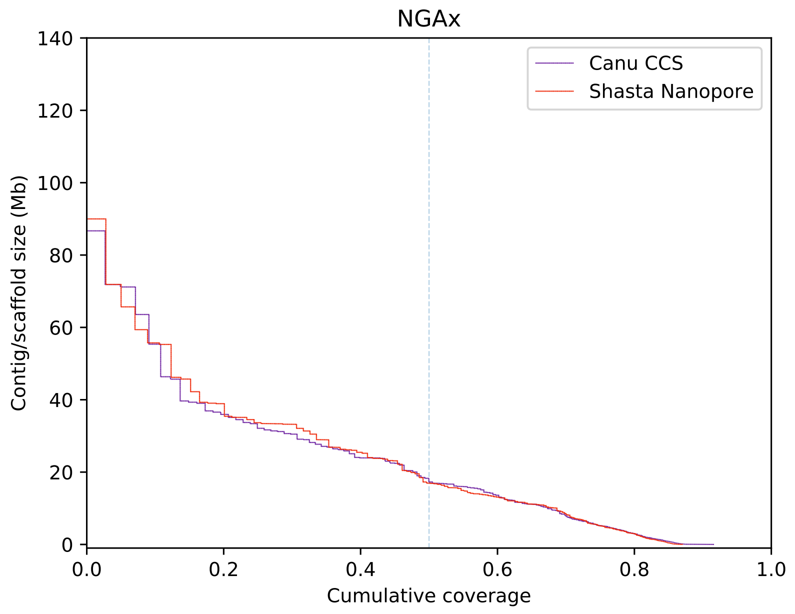
Sample	Assembler	Total disagreements	Consensus disagreements
CHM13	PacBio-HiFi	8469	594
	Shasta	1073	380

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

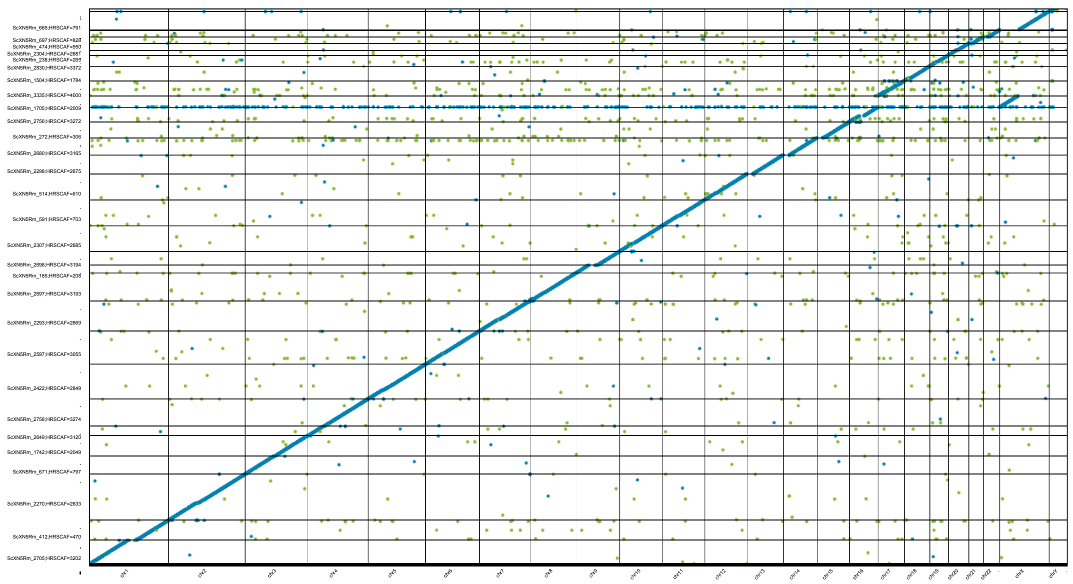
Sample	Sequencing Platform	Method		Percentage errors			
		Assembler	Polisher	Balanced	Identity	Deletion	Insertion
CHM13 Chr-X	PacBio HiFi	Canu	Racon	0.008%	0.001%	0.004%	0.003%
	Nanopore	Shasta	MarginPolish & HELEN	0.064%	0.006%	0.036%	0.022%



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



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



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

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

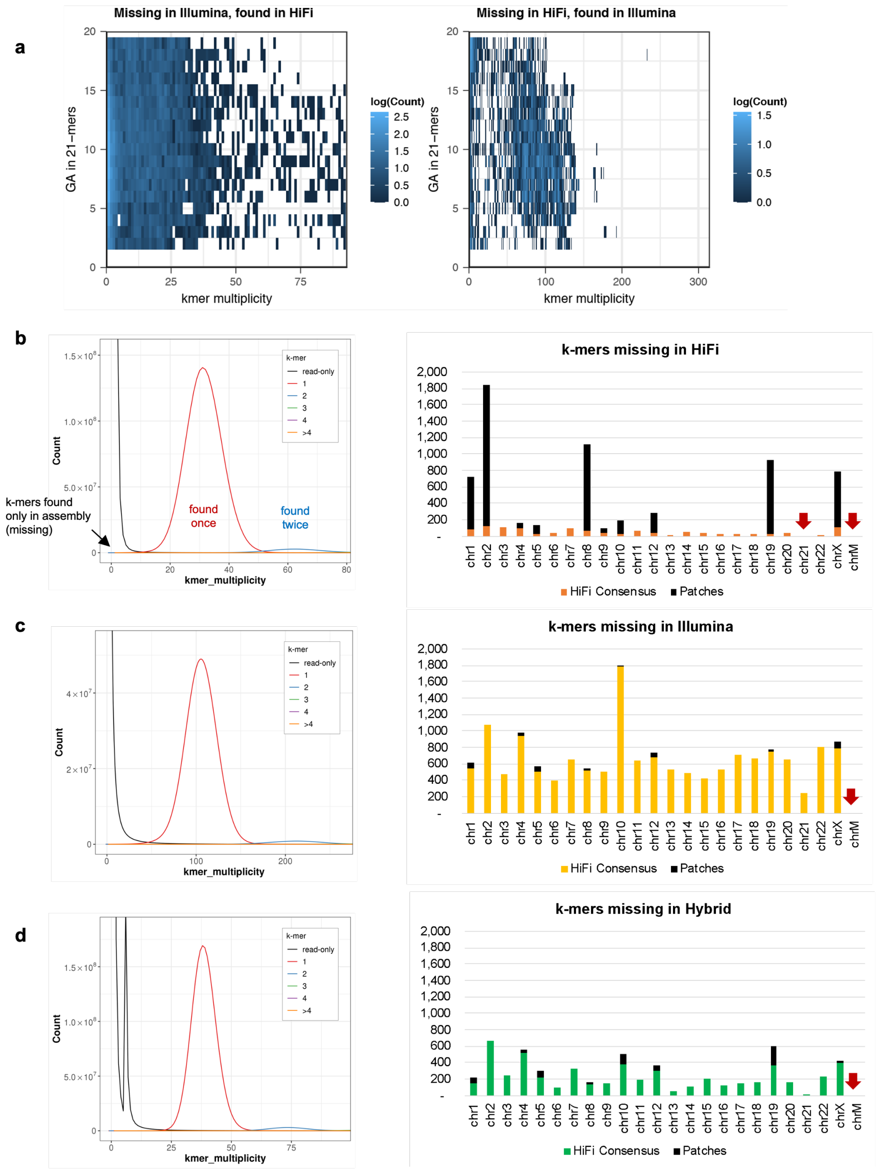
Sample	# contigs	Total length	N50	NG50	# mis-assemblies	# scaffold gap extensive mis-assemblies	Genome fraction (%)	# mismatches per 100 kbp	# indels per 100 kbp	Total aligned length	NA50	NGA50
GM24143	1,184	2,802,523,049	129,960,437	128,216,303	1,466	4	95.027	128.28	142.79	2,792,775,664	20,657,530	16,966,477
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 C

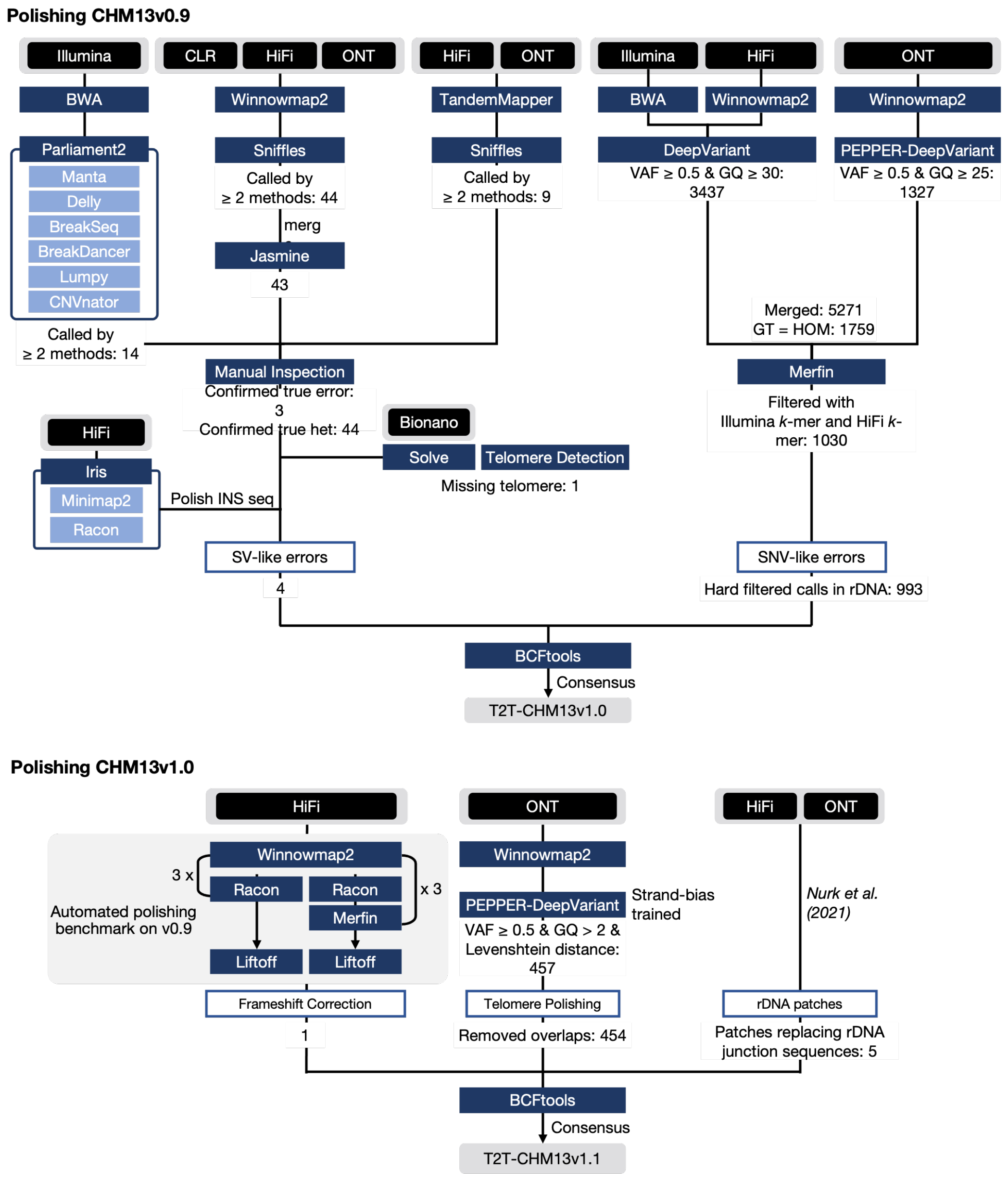
Appendix C: Supplementary information for validation and polishing of the first complete human genome

Preamble

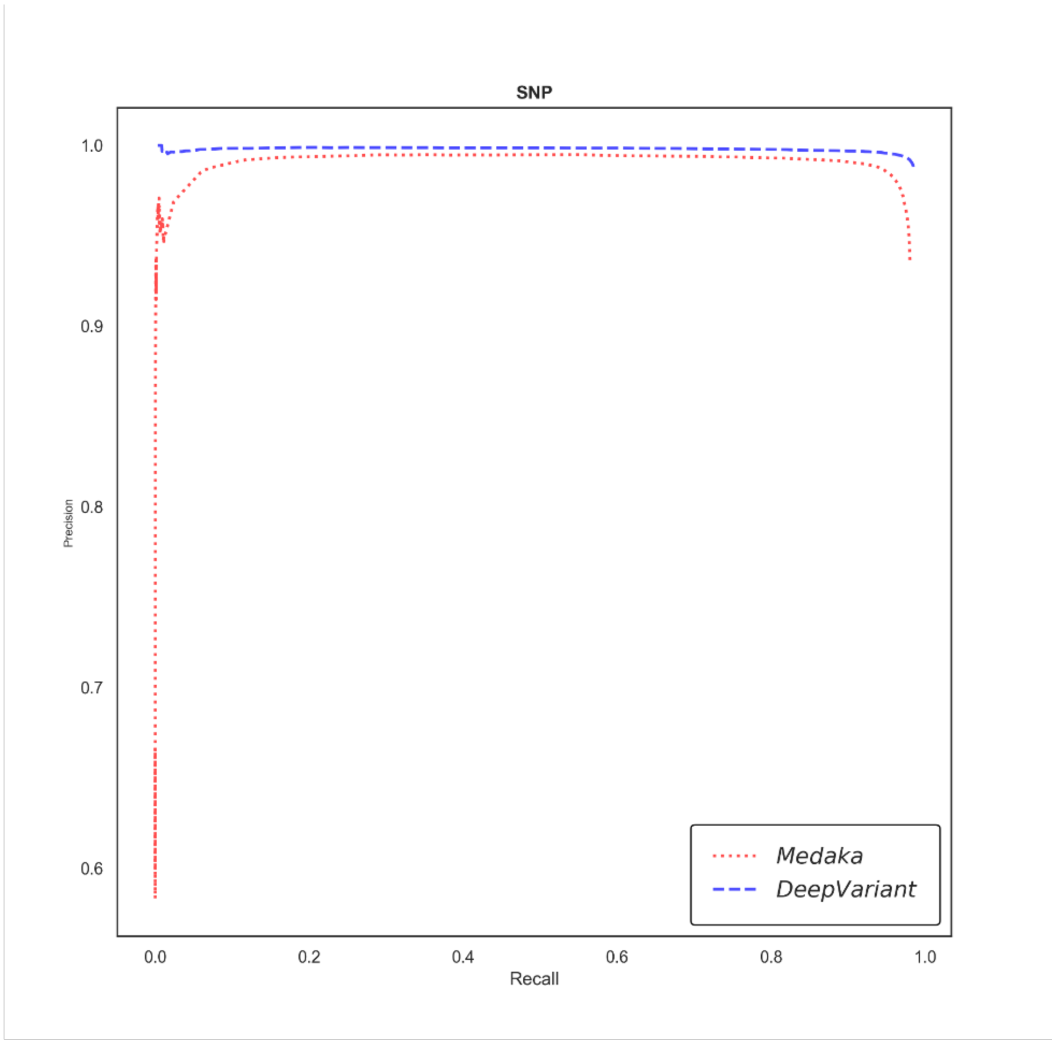
Supplementary Results



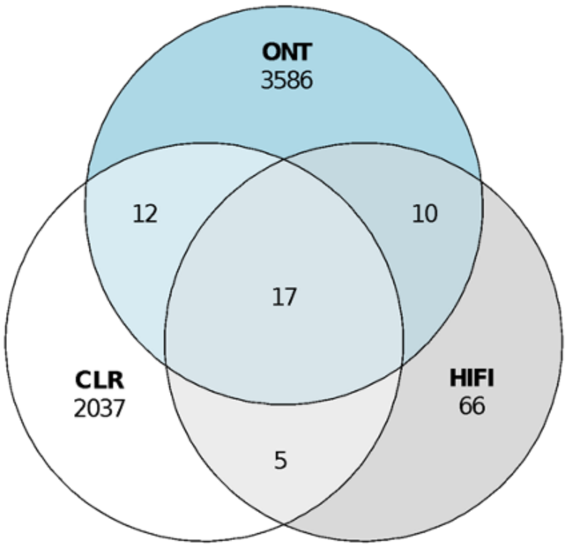
Supplementary Figure C.1: Sequencing biases observed in missing kmers. a, missing k-mers with its GA composition. b-d, v0.9 assembly and k-mer copy number spectrum from HiFi, Illumina, and hybrid k-mer sets (left) and per-chromosome missing (likely error) k-mer counts from the HiFi derived consensus or patches (right). Most missing k-mers in HiFi overlapped sequences from patched regions. No missing k-mer was found on Chromosomes indicated with red arrows.



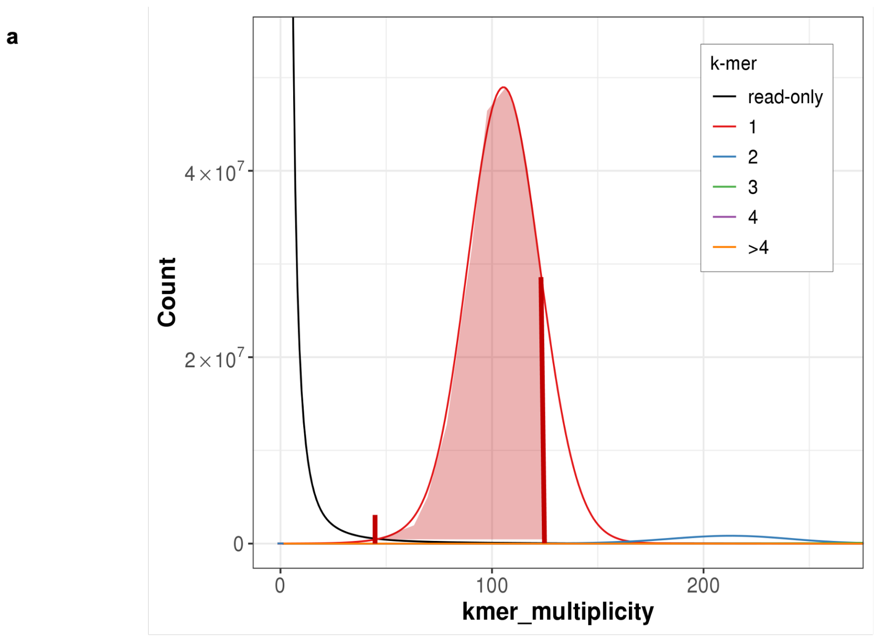
Supplementary Figure C.2: Error detection and polishing pipeline. A detailed overview of the polishing pipeline along with the number of errors identified and polished at each step. Additionally, data type and polishing tools utilized are highlighted. Illumina, 100X PCR-free library Illumina reads; HiFi, 35x PacBio HiFi reads; ONT, 120x Oxford Nanopore reads.

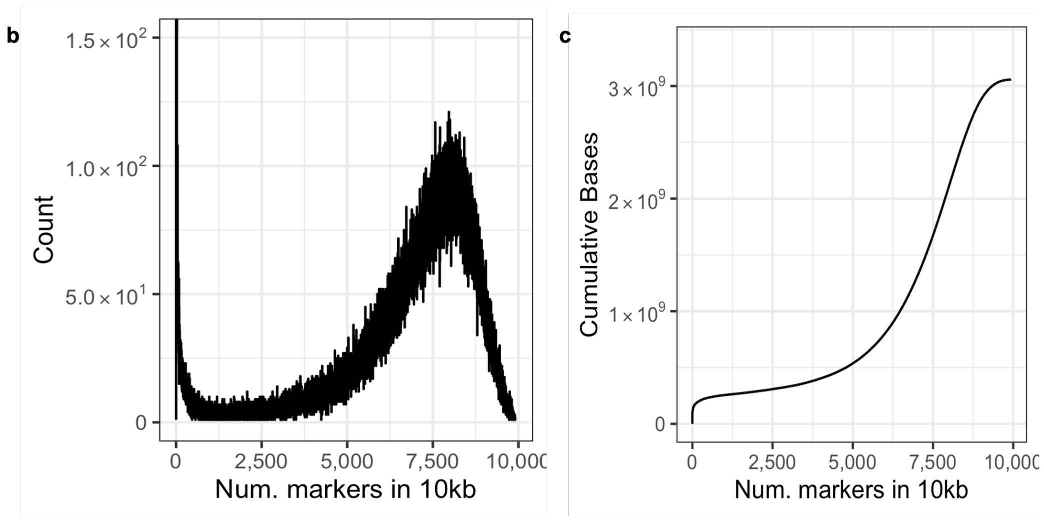


Supplementary Figure C.3: SNV-like error filtering. ONT PEPPER-DeepVariant SNP call were more reliable with both higher precision and recall over Medaka.



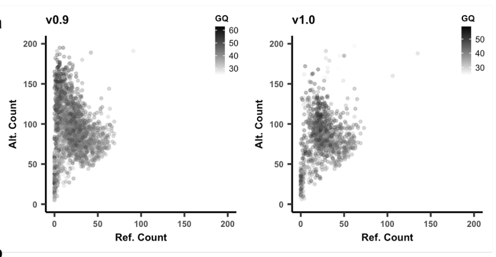
Supplementary Figure C.4: Number of SV-like errors called from long-read platforms.

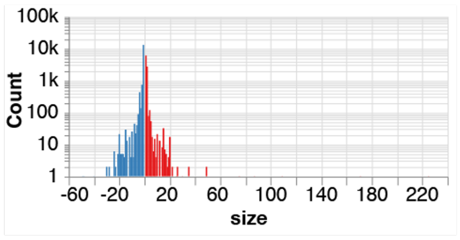




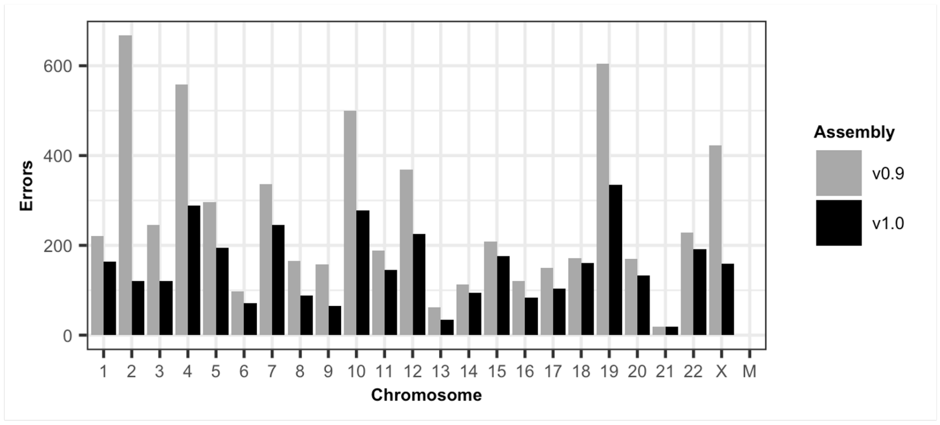
Supplementary Figure C.5: Globally unique single-copy kmers used for marker assisted alignment. a. Range of k-mer counts defined as ‘single-copy’ markers from Illumina reads and in the assembly. The cutoffs were chosen to minimize inclusion of low-frequency erroneous kmers and 2-copy k-mers. b. Number of markers in every 10 kb window. c. Cumulative number of bases covered by the number of markers in each 10 kb window.

a

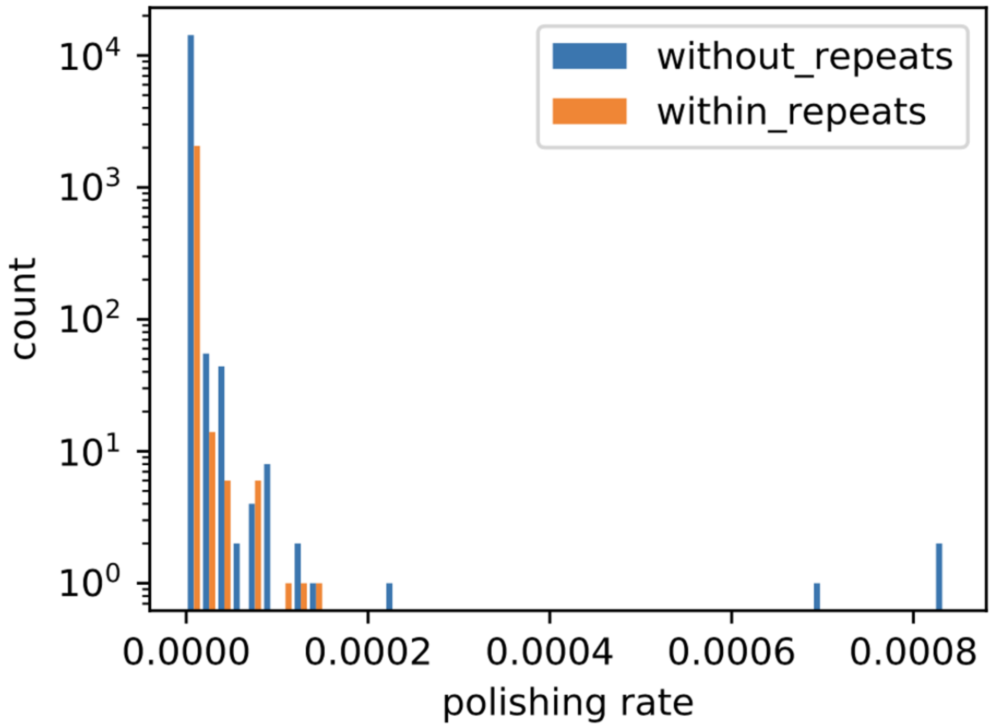




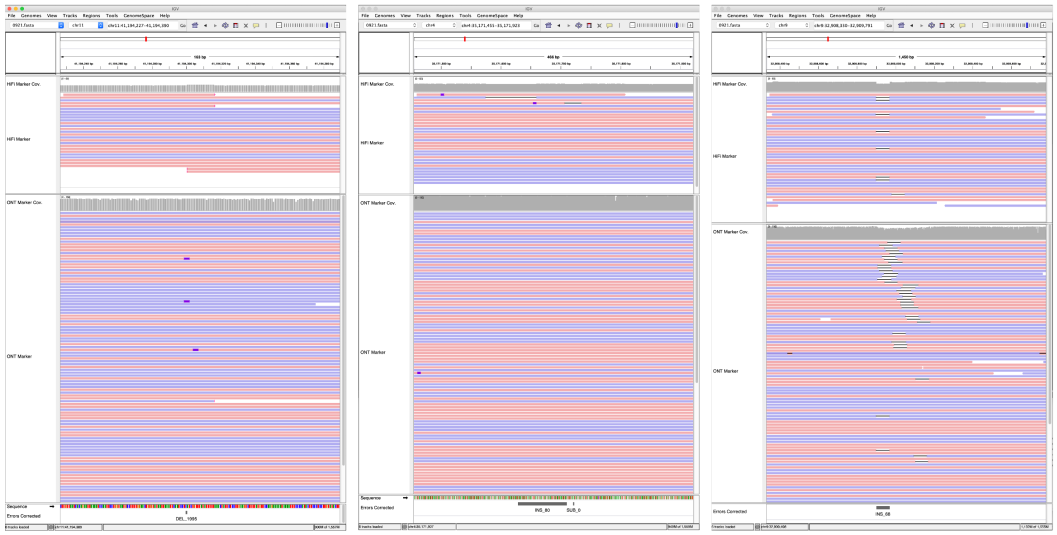
b



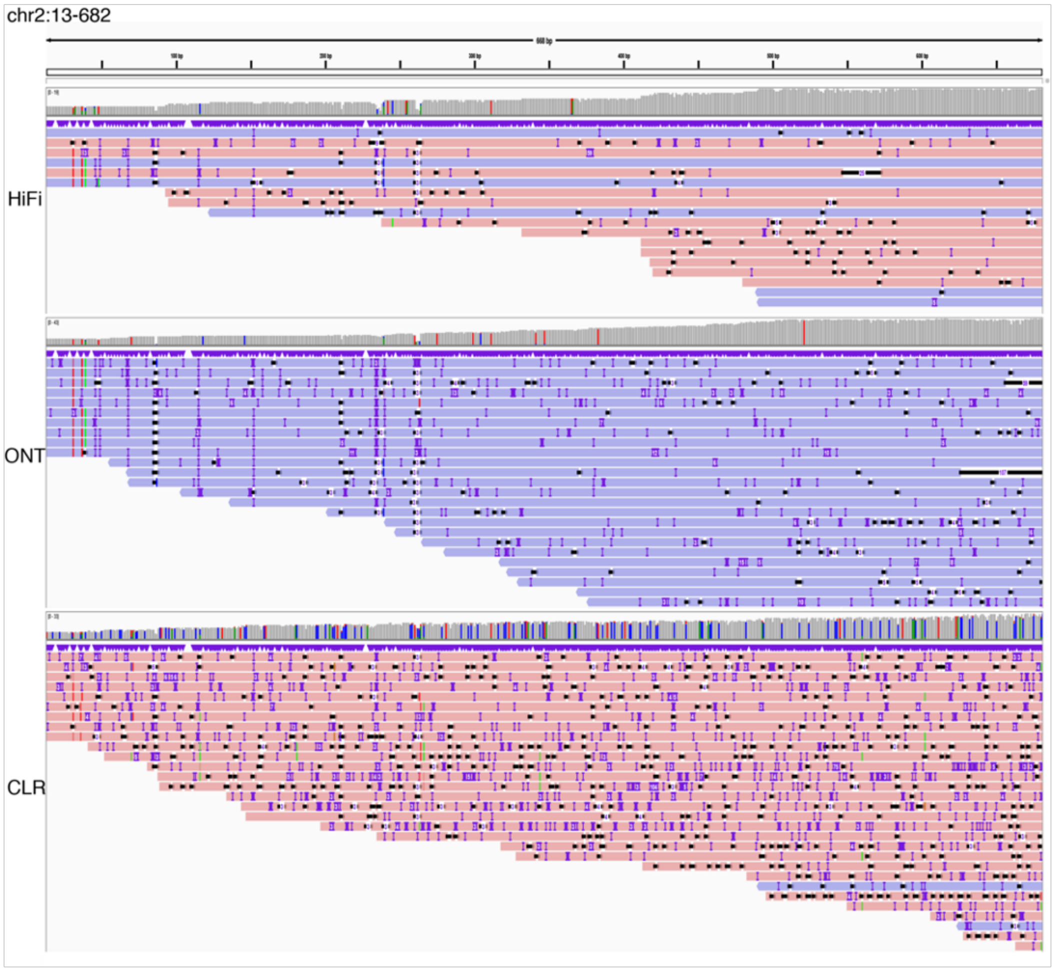
Supplementary Figure C.6: Post-polishing evaluation. a. Left, genotype quality and number of reads supporting the reference and alternate alleles from the combined Illumina-hifi hybrid and ONT homozygous variant calls, with AF > 0.5. Right, balanced insertion (red) and deletion (blue) length distribution from the Illumina-HiFi hybrid DeepVariant heterozygous calls in v1.0. b. Number of errors detected in each chromosome, before and after polishing.



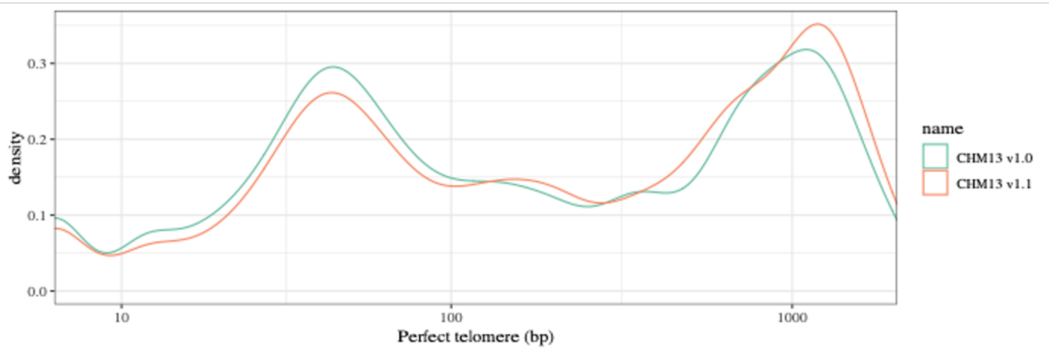
Supplementary Figure C.7: Polishing inside and outside of repeats. The distribution of v0.9 polishing rates within and without repeats.



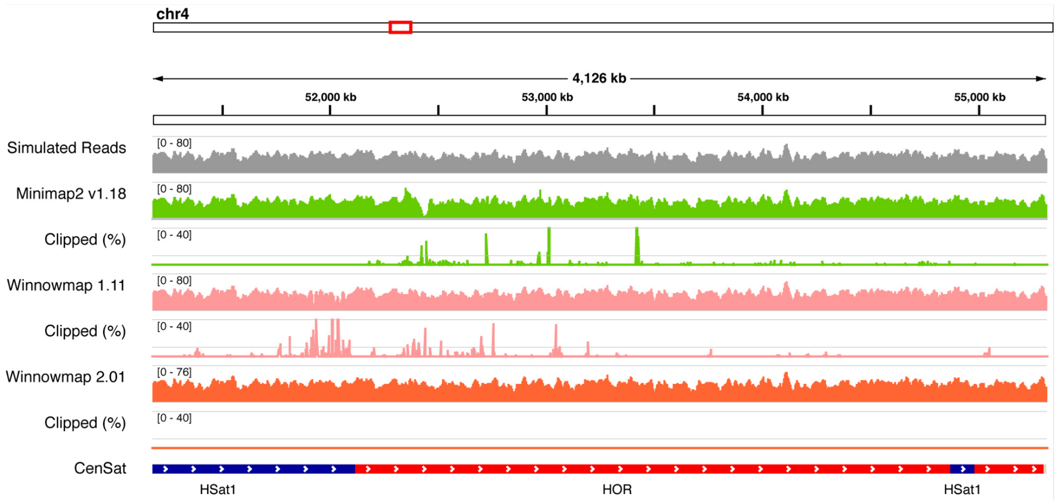
Supplementary Figure C.8: Three SV-like errors corrected. HiFi and ONT marker assisted alignments, post correction of the 3 large SV-like edits visualized with IGV. HiFi coverage track is shown in data range up to 60, ONT up to 150. Clipped reads are flagged for >100bp. INDELs smaller than 10bp are not shown. Reads are colored by strands; positive in red and negative in blue.



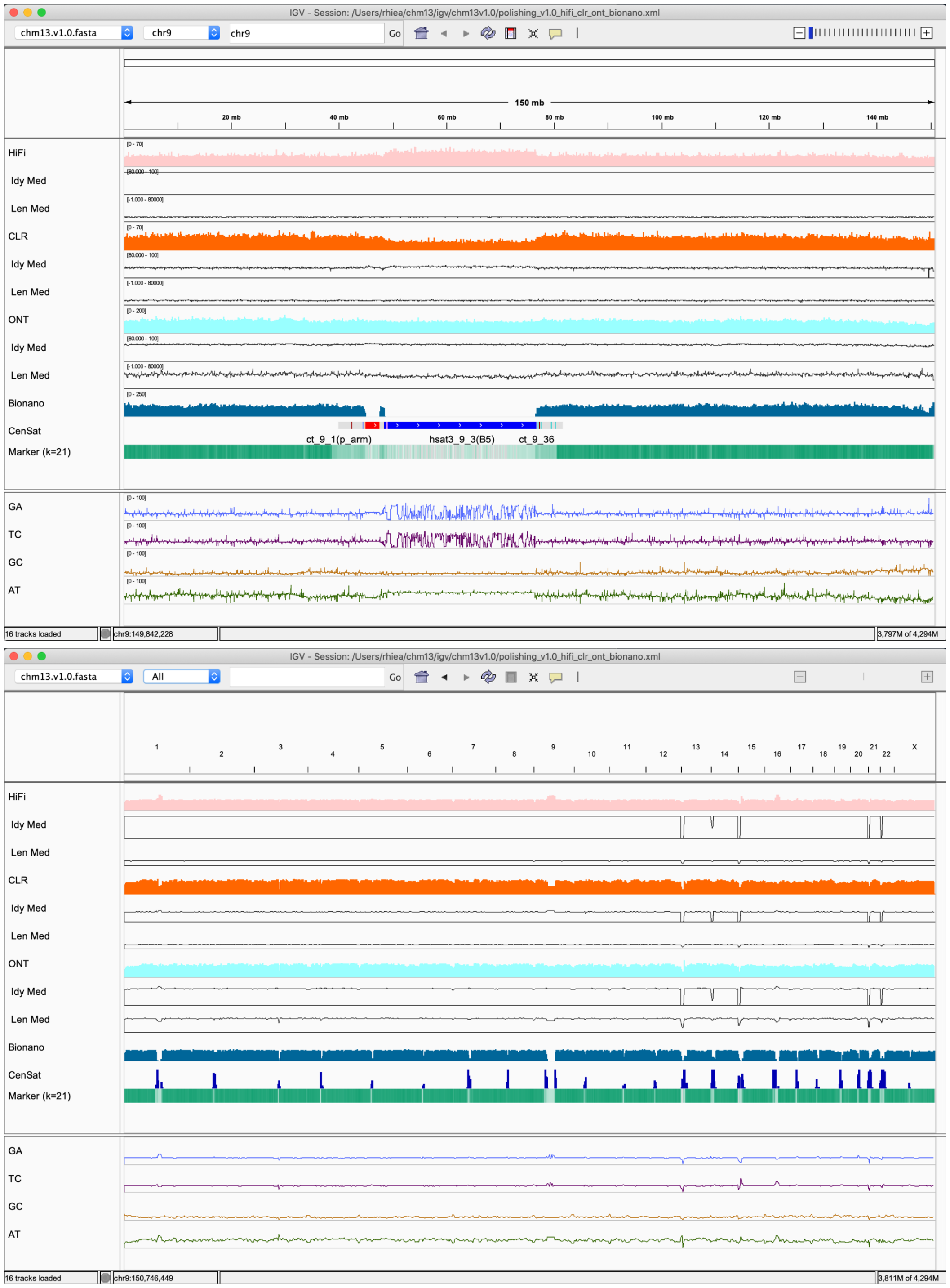
Supplementary Figure C.9: An illustration of chr 2 telomere sequence reads from HiFi, ONT and CLR platform.



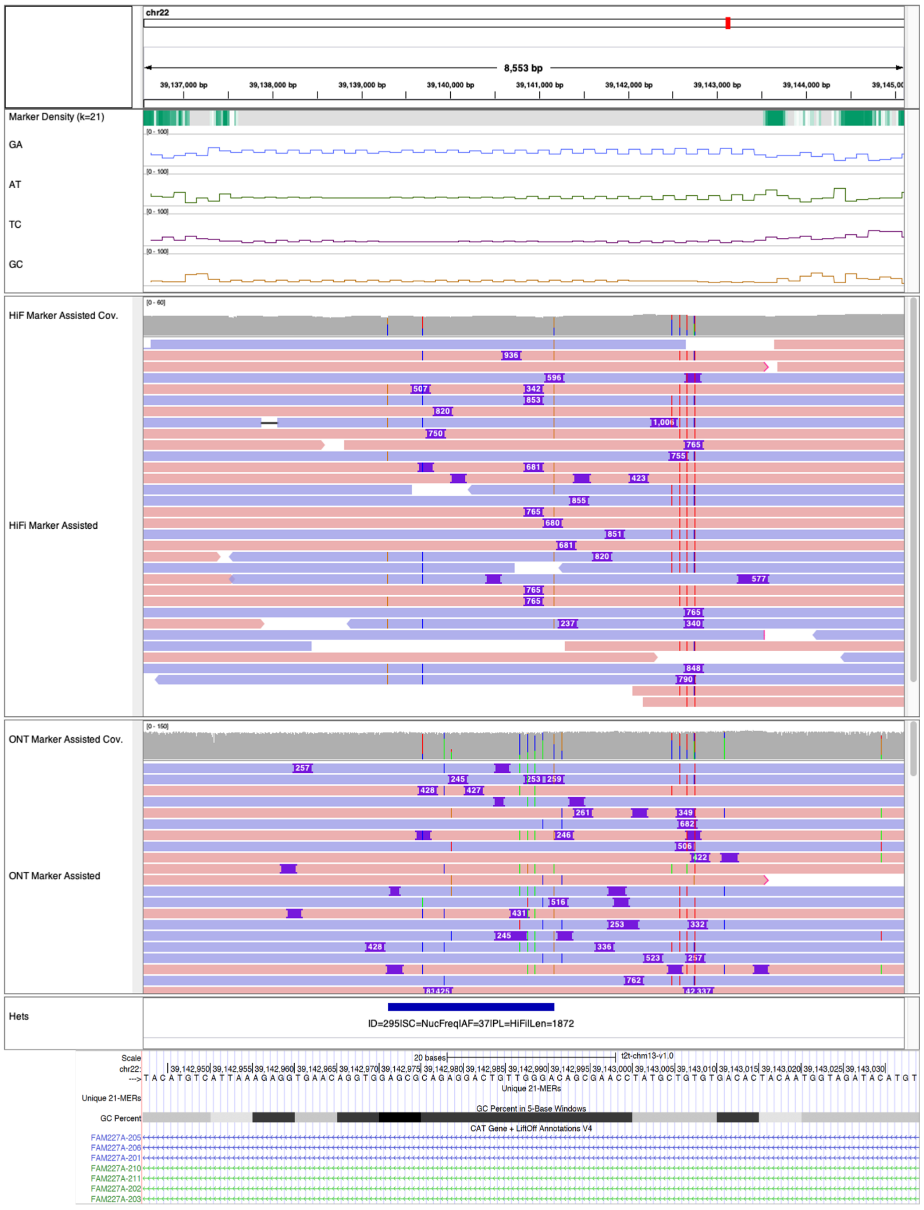
Supplementary Figure C.10: Distribution of maximum perfect match to the canonical k-mer observed at each position in the telomere before (CHM13 v1.0) and after (CHM13 v1.1) polishing the telomeres.



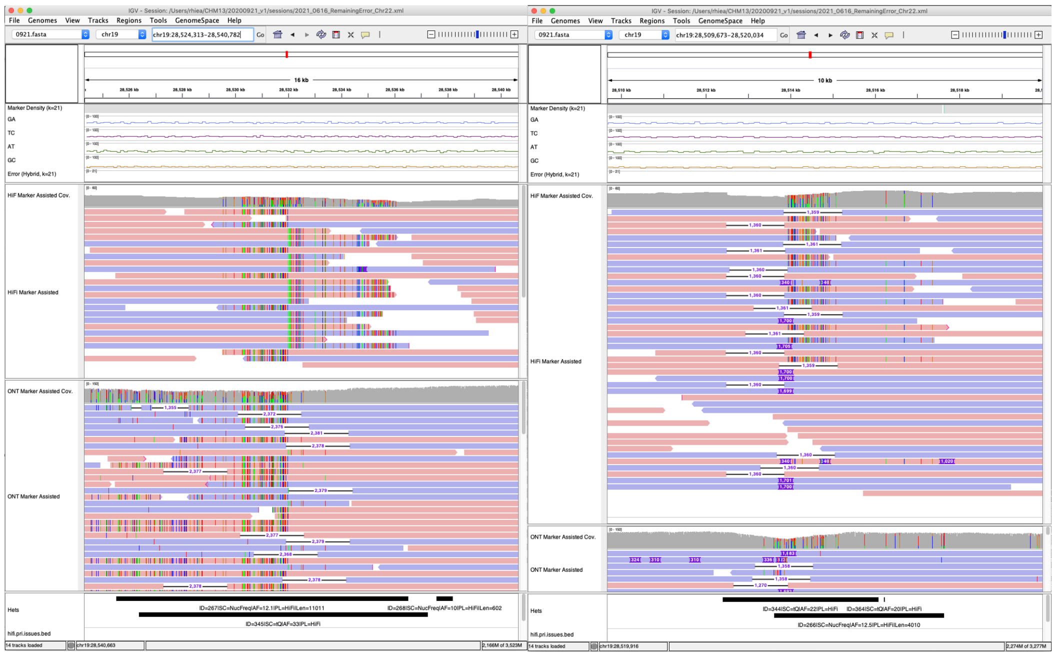
Supplementary Figure C.11: Mapping biases found and corrected. On simulated HiFi reads, we found excessive clippings in highly identical satellite repeats in Minimap and Winnowmap by the time of evaluation. We have addressed this issue in Winnowmap 2.01+. Clipped (%) indicates the percentage of reads clipped in every 1024 bp window, shown in 0 40% range with a midline of 10%.



Supplementary Figure C.12: HiFi, CLR, ONT read coverage, alignment identity, and read length from Winnowmap2 v2.01 alignments and Bionano DLE-1 molecule coverage from Bionano Solve. Upper panel shows a zoomed in region of Chromosome 9, while the upper panel shows the whole-genome alignment view. HiFi, CLR, ONT, and Bionano coverage are shown up to 70x, 70x, 200x, and 250x, respectively. Median read identity in every 1024 bp is shown in 80-100% range. Median read length in every 1024 bp is shown in 0-100kb range. Read identity was the worst in CLR, and between HiFi and ONT. Bionano molecules were lacking coverage in most of the centromeric repeats.



Supplementary Figure C.13: Collapsed simple tandem repeat. The collapse in the Intronic sequences of gene FAM227A was undetected, due to the variable insertion breakpoints and insertion length in the HiFi and ONT alignments. The panels above the alignments show marker density and percent microsatellites (GA / AT / TC / GC) in each 64 bp window, which indicates this region is highly repetitive with GA enriched sequences, which later alternates with AT enriched sequences.


Supplementary Figure C.14: Chimeric junction of two haplotypes. In the shown above regions, both HiFi and ONT reads indicate that the consensus has a chimeric junction of the two haplotypes.

	PacBio HiFi	Illumina	Hybrid
QV			
V0.9	69.68	66.09	70.22
V1.0	69.88	67.28	72.62
V1.1	69.80	67.86	73.94
K-mers found only in assembly (Error k-mers)			
V0.9	6,881	15,723	6,073
V1.0	6,581	11,961	3,496
V1.1	6,724	10,497	2,591
K-mers found in both assembly and reads			
V0.9	3,045,438,411	3,045,438,411	3,045,438,411
V1.0	3,045,440,942	3,045,440,942	3,045,440,942

Supplementary Table C.1: K-mer based consensus quality evaluation. From each sequencing dataset and assembly versions, 21-mers were collected and compared with Merqury.

Query Assembly	True Negatives	False Positives	True Positives	False Negatives
CHM13v1.0	69123	0	16	611
Racon	69101	22	19	608
Racon+Merfin	69123	0	14	613

Supplementary Table C.2: Correcting invalid ORFs via polishing. This considers the classification of GRCh38 transcripts with valid ORFs. True/False Positives/Negatives for each query assembly were assessed as follows: If a transcript has a valid ORF in the CHM13v0.9 assembly and it remains valid in the query assembly, it is a "True Negative". If a transcript has a valid ORF in the CHM13v0.9 assembly but it has an invalid ORF in the query assembly, it is a "False Positive". If a transcript has an invalid ORF in the CHM13v0.9 assembly and it remains invalid in the query assembly, it is a "False Negative", and if a transcript has an invalid ORF in the CHM13v0.9 assembly but it has a valid ORF in the query assembly, it is a "True Positive". Only transcripts that mapped to all assemblies were considered.

CHM13 v1.0			CHM13 v1.1			
Chr.	Start	End	Chr.	Start	End	Reason
11	54243938	54285164	11	54258631	54260218	Low consensus quality
16	36735111	36775613	16	36753902	36757498	Low consensus quality
n/a			15	3642490	3643193	GA sequencing bias in model rDNA sequence
n/a			15	3732691	3732910	GA sequencing bias in model rDNA sequence
22	39136546	39145169	22	39107778	39116401	Collapsed low complexity sequence
19	28509243	28519604	19	28509246	28519607	Chimeric consensus of two haplotypes
19	28527178	28538917	19	28527181	28538920	Chimeric consensus of two haplotypes

Supplementary Table C.3: Remaining issues identified from both PacBio HiFi and ONT

Cause of low coverage	Num. of regions	(%)
Likely low consensus qual.	11	5.0%
Low consensus qual.	30	13.8%
AT biases	7	3.2%
GA or TC biases	170	78.0%

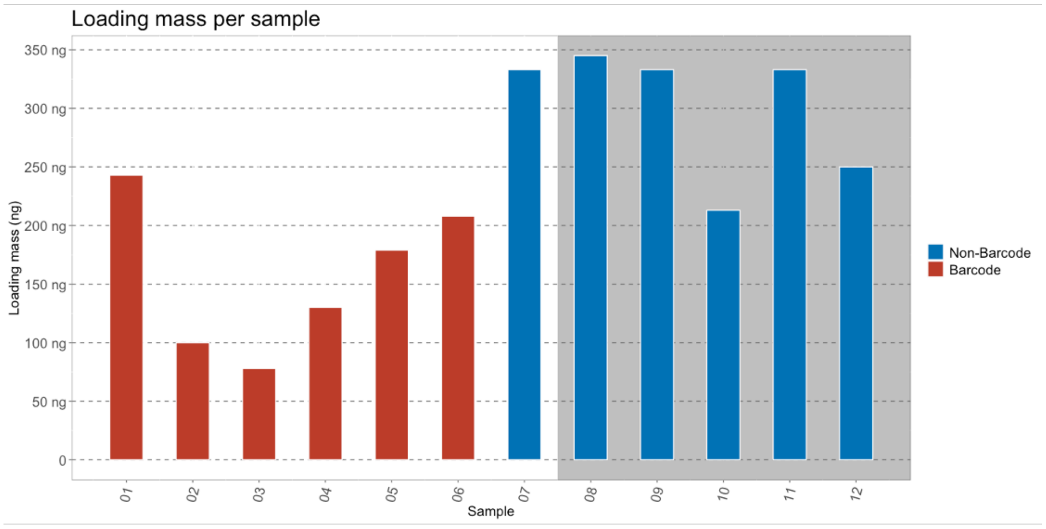
Supplementary Table C.4: Low coverage regions detected only from HiFi alignments. Regions with $<$7x Winnowmap primary read alignments were collected and categorized given the mapping quality, alignment identity, and sequence context (% microsatellites within 10 kb).

Appendix D

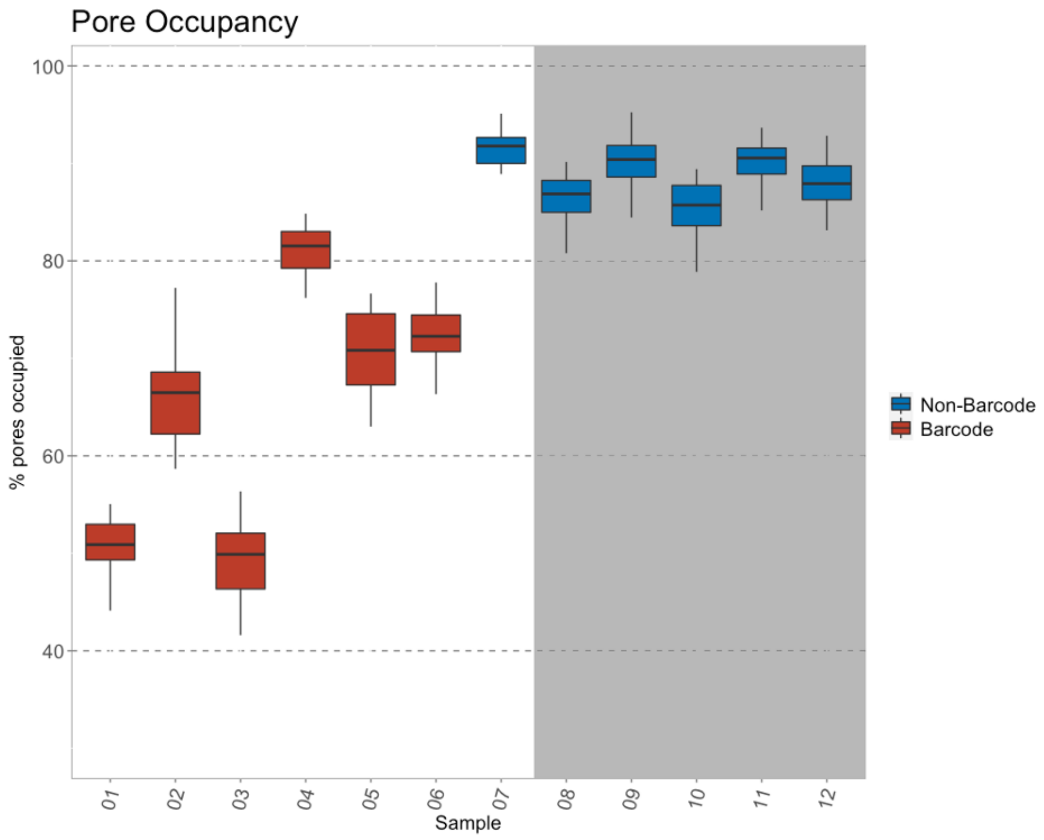
Appendix A: Supplementary information for ultra-rapid whole genome nanopore sequencing in a critical care setting

Preamble

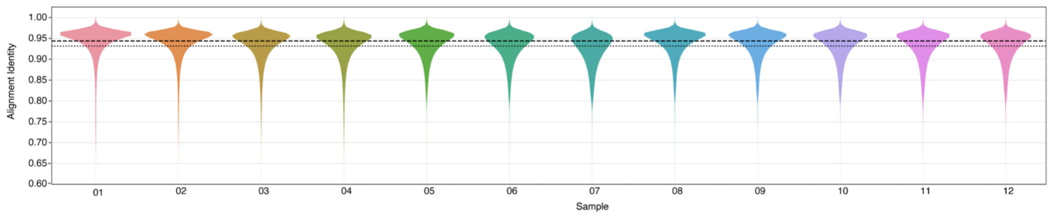
Supplementary Figures



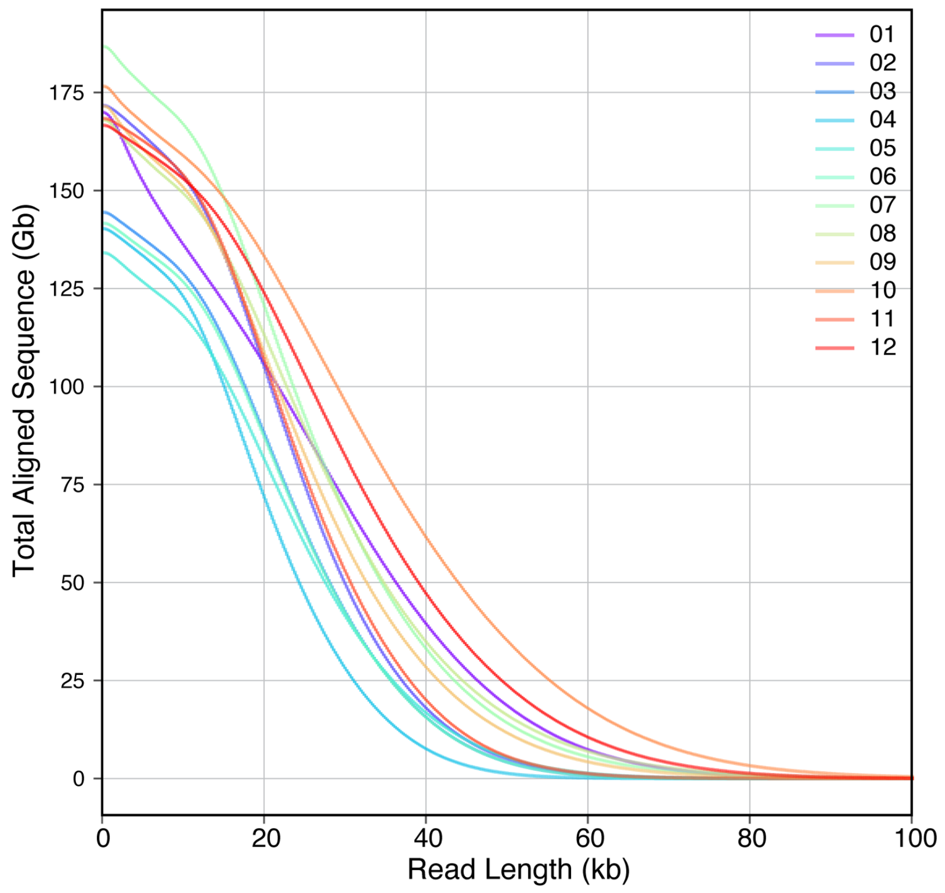
Supplementary Figure D.1: Ligating barcodes reduces the final yield of the sequencing library. The mean DNA loading mass per flow cell of barcoded libraries (red) was 155 ng (78 ng—243 ng) compared to non-barcoded libraries (blue) 333 ng (208 ng—345 ng). This difference was observed in phase one (white background), and it was decided to continue with the non-barcoding protocol in phase 2 (gray background).



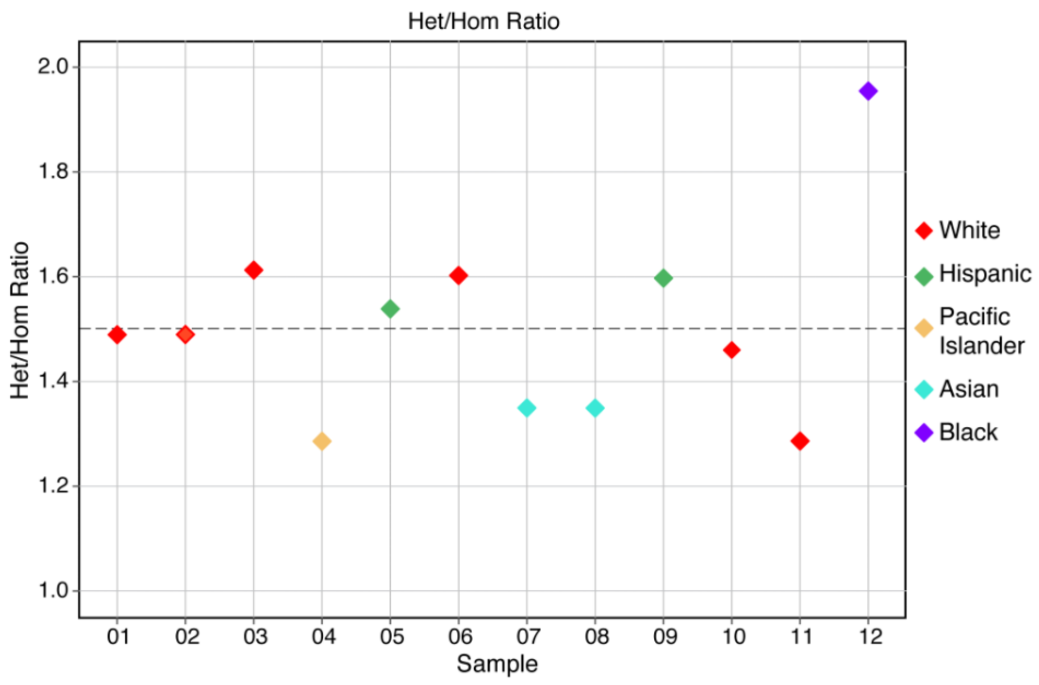
Supplementary Figure D.2: Eliminating barcoding results in higher pore occupancy. Barcoded libraries (red) have an average pore occupancy of 64% compared to non-barcoded libraries (Blue) of 82%. This difference was observed in phase one (white background), and it was decided to continue with the non-barcoding protocol in phase 2 (gray background). This increased pore occupancy is suspected to be a direct result of the increased yield of library in the non-barcoded samples



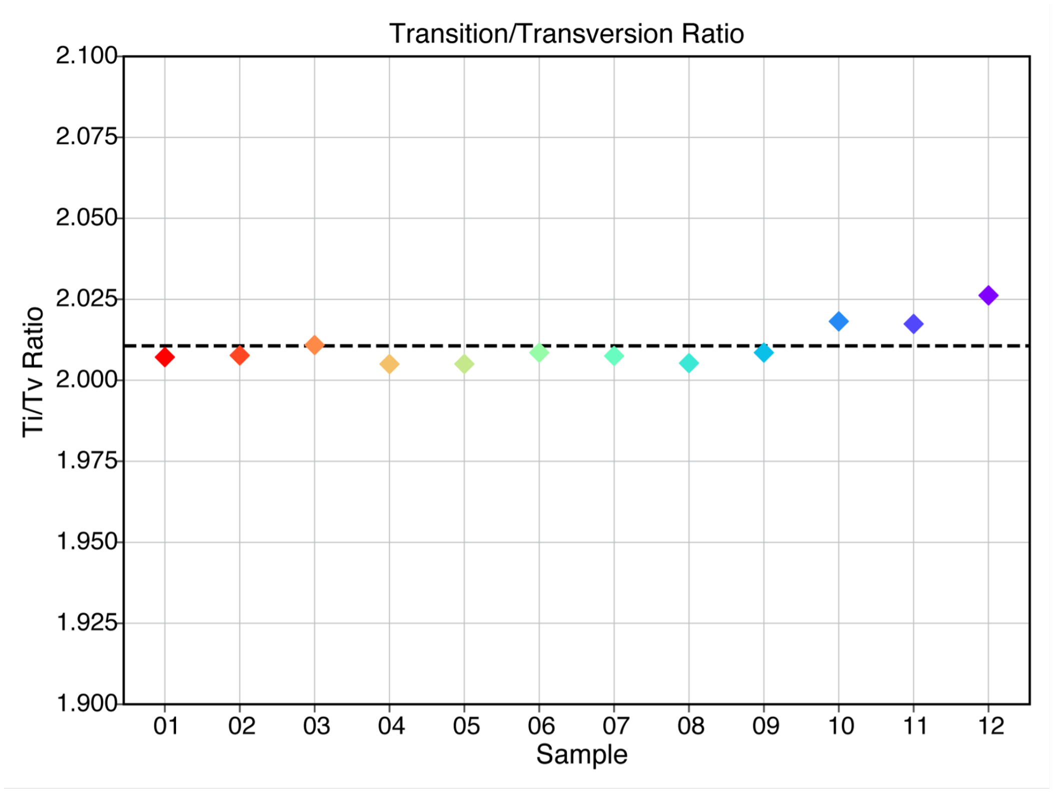
Supplementary Figure D.3: Alignment identities against GRCh37. Median of 0.944 is shown by the dashed line and the mean of 0.931 is shown by the dotted line.



Supplementary Figure D.4: NGx plot: Total aligned sequence (Gb) for the samples as a function of the read length.



Supplementary Figure D.5: Heterozygous/Homozygous ratio. Diamonds are colored based on the patient's ethnicity. The dashed line indicates a mean of 1.5



Supplementary Figure D.6: Transition/Transversion ratio for the SNP calls across all the patient samples. The dashed line indicates a mean of 2.0

Bibliography

[1] S Aganezov. A complete human reference genome improves variant calling for population and clinical genomics. *bioRxiv (to appear)*, 2021.

[2] Sergey Aganezov, Stephanie M Yan, Daniela C Soto, Melanie Kirsche, Samantha Zarate, Pavel Avdeyev, Dylan J Taylor, Kishwar Shafin, Alaina Shumate, Chunlin Xiao, et al. A complete reference genome improves analysis of human genetic variation. *bioRxiv*, 2021.

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

[4] N Altemose. Genetic and epigenetic maps of endogenous human centromeres. *bioRxiv (to appear)*, 2021.

[5] Nicolas Altemose, Glennis A Logsdon, Andrey V Bzikadze, Pragya Sidhwani, Sasha A Langley, Gina V Caldas, Savannah J Hoyt, Lev Uralsky, Fedor D Ryabov, Colin J Shew, et al. Complete genomic and epigenetic maps of human centromeres. *bioRxiv*, 2021.

[6] 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.

[7] Gunjan Baid, Daniel E Cook, Kishwar Shafin, Taedong Yun, Felipe Llinares-Lopez, Quentin Berthet, Aaron M Wenger, William J Rowell, Maria Nattestad, Howard Yang, et al. Deepconsensus: Gap-aware sequence transformers for sequence correction. *bioRxiv*, 2021.

[8] 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.

[9] N Baran, A Lapidot, and H Manor. Formation of DNA triplexes accounts for arrests of DNA synthesis at d(TC)n and d(GA)n tracts, 1991.

[10] 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.

[11] 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.

[12] Blue Shield of California and News Center. Blue shield of california becomes first health plan in u.s. to cover cost of rapid whole genome sequencing for critically ill children. https://news.blueshieldca.com/2020/03/09/RADY-genomics. Accessed: 2021-5-29.

[13] Daniel M Bornman, Mark E Hester, Jared M Schuetter, Manjula D Kasoji, Angela Minard-Smith, Curt A Barden, Scott C Nelson, Gene D Godbold, Christine H Baker, Boyu Yang, et al. Short-read, high-throughput sequencing technology for str genotyping. *BioTechniques. Rapid dispatches*, 2012:1, 2012.

[14] 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.

[15] 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.

[16] Henry Brinkerhoff, Albert SW Kang, Jingqian Liu, Aleksei Aksimentiev, and Cees Dekker. Multiple rereads of single proteins at single–amino acid resolution using nanopores. *Science*, 374(6574):1509–1513, 2021.

[17] 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.

[18] Patrick O Brown and David Botstein. Exploring the new world of the genome with dna microarrays. *Nature genetics*, 21(1):33–37, 1999.

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

[20] Jillian G Buchan, Shana White, Ruchi Joshi, and Euan A Ashley. Rapid genome sequencing in the critically ill. *Clin. Chem.*, 65(6):723–726, June 2019.

[21] Yue Cao, Thomas Andrew Geddes, Jean Yee Hwa Yang, and Pengyi Yang. Ensemble deep learning in bioinformatics. *Nature Machine Intelligence*, 2(9):500–508, 2020.

[22] Mark J P Chaisson, Ashley D Sanders, Xuefang Zhao, Ankit Malhotra, David Porubsky, Tobias Rausch, Eugene J Gardner, Oscar L Rodriguez, Li Guo, Ryan L Collins, and Others. Multi-platform discovery of haplotype-resolved structural variation in human genomes. *Nat. Commun.*, 10(1):1–16, 2019.

[23] Mark JP Chaisson, John Huddleston, Megan Y Dennis, Peter H Sudmant, Maika Malig, Fereydoun Hormozdiari, Francesca Antonacci, Urvashi Surti, Richard Sandstrom, Matthew Boitano, et al. Resolving the complexity of the human genome using single-molecule sequencing. *Nature*, 517(7536):608–611, 2015.

[24] 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.

[25] Yen-Chun Chen, Tsunglin Liu, Chun-Hui Yu, Tzen-Yuh Chiang, and Chi-Chuan Hwang. Effects of GC bias in next-generation-sequencing data on de novo genome assembly. *PLoS One*, 8(4):e62856, April 2013.

[26] 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.

[27] Haoyu Cheng, Gregory T Concepcion, Xiaowen Feng, Haowen Zhang, and Heng Li. Haplotype-resolved de novo assembly using phased assembly graphs with hifiasm. *Nat. Methods*, February 2021.

[28] 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.

[29] Chen-Shan Chin, Paul Peluso, Fritz J Sedlazeck, Maria Nattestad, Gregory T

Concepcion, Alicia Clum, Christopher Dunn, Ronan O'Malley, Rosa Figueroa-Balderas, Abraham Morales-Cruz, et al. Phased diploid genome assembly with single-molecule real-time sequencing. *Nature methods*, 13(12):1050, 2016.

[30] 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.

[31] Michelle M Clark, Amber Hildreth, Sergey Batalov, Yan Ding, Shimul Chowdhury, Kelly Watkins, Katarzyna Ellsworth, Brandon Camp, Cyrielle I Kint, Calum Yacoubian, Lauge Farnaes, Matthew N Bainbridge, Curtis Beebe, Joshua J A Braun, Margaret Bray, Jeanne Carroll, Julie A Cakici, Sara A Caylor, Christina Clarke, Mitchell P Creed, Jennifer Friedman, Alison Frith, Richard Gain, Mary Gaughran, Shauna George, Sheldon Gilmer, Joseph Gleeson, Jeremy Gore, Haiying Grunenwald, Raymond L Hovey, Marie L Janes, Kejia Lin, Paul D McDonagh, Kyle McBride, Patrick Mulrooney, Shareef Nahas, Daeheon Oh, Albert Oriol, Laura Puckett, Zia Rady, Martin G Reese, Julie Ryu, Lisa Salz, Erica Sanford, Lawrence Stewart, Nathaly Sweeney, Mari Tokita, Luca Van Der Kraan, Sarah White, Kristen Wigby, Brett Williams, Terence Wong, Meredith S Wright, Catherine Yamada, Peter Schols, John Reynders, Kevin Hall, David Dimmock, Narayanan Veeraraghavan, Thomas Defay, and Stephen F Kingsmore. Diagnosis of genetic diseases in seriously ill children by rapid whole-genome sequencing and automated phenotyping and interpretation. *Sci. Transl. Med.*, 11(489), April 2019.

[32] John G Cleary, Ross Braithwaite, Kurt Gaastra, Brian S Hilbush, Stuart Inglis, Sean A Irvine, Alan Jackson, Richard Littin, Sahar Nohzadeh-Malakshah, Mehul Rathod, et al. Joint variant and de novo mutation identification on pedigrees from high-throughput sequencing data. *Journal of Computational Biology*, 21(6):405–419, 2014.

[33] 1000 Genomes Project Consortium et al. A map of human genome variation from population scale sequencing. *Nature*, 467(7319):1061, 2010.

[34] 1000 Genomes Project Consortium et al. An integrated map of genetic variation from 1,092 human genomes. *Nature*, 491(7422):56, 2012.

[35] 1000 Genomes Project Consortium et al. A global reference for human genetic variation. *Nature*, 526(7571):68, 2015.

[36] Matei David, Lewis Jonathan Dursi, Delia Yao, Paul C Boutros, and Jared T Simpson. Nanocall: an open source basecaller for oxford nanopore sequencing data. *Bioinformatics*, 33(1):49–55, 2017.

[37] Wouter De Coster, Matthias H Weissensteiner, and Fritz J Sedlazeck. Towards population-scale long-read sequencing. *Nat. Rev. Genet.*, May 2021.

[38] Wouter De Coster, Matthias H Weissensteiner, and Fritz J Sedlazeck. Towards population-scale long-read sequencing. *Nature Reviews Genetics*, pages 1–16, 2021.

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

[40] Frederick E Dewey, Rong Chen, Sergio P Cordero, Kelly E Ormond, Colleen Caleshu, Konrad J Karczewski, Michelle Whirl-Carrillo, Matthew T Wheeler, Joel T Dudley, Jake K Byrnes, Omar E Cornejo, Joshua W Knowles, Mark Woon, Katrin Sangkuhl, Li Gong, Caroline F Thorn, Joan M Hebert, Emidio Capriotti, Sean P David, Aleksandra Pavlovic, Anne West, Joseph V Thakuria, Madeleine P Ball, Alexander W Zaranek, Heidi L Rehm, George M Church, John S West, Carlos D Bustamante, Michael Snyder, Russ B Altman, Teri E Klein, Atul J Butte, and Euan A Ashley. Phased whole-genome genetic risk in a family quartet using a major allele reference sequence. *PLoS Genet.*, 7(9):e1002280, September 2011.

[41] Frederick E Dewey, Megan E Grove, Cuiping Pan, Benjamin A Goldstein, Jonathan A Bernstein, Hassan Chaib, Jason D Merker, Rachel L Goldfeder, Gregory M Enns, Sean P David, Neda Pakdaman, Kelly E Ormond, Colleen Caleshu, Kerry Kingham, Teri E Klein, Michelle Whirl-Carrillo, Kenneth Sakamoto, Matthew T Wheeler, Atul J Butte, James M Ford, Linda Boxer, John P A Ioannidis, Alan C Yeung, Russ B Altman, Themistocles L Assimes, Michael Snyder, Euan A Ashley, and Thomas Quertermous. Clinical interpretation and implications of whole-genome sequencing. *JAMA*, 311(10):1035–1045, March 2014.

[42] Egor Dolzhenko, Mark F Bennett, Phillip A Richmond, Brett Trost, Sai Chen, Joke J F A van Vugt, Charlotte Nguyen, Giuseppe Narzisi, Vladimir G Gainullin, Andrew M Gross, Bryan R Lajoie, Ryan J Taft, Wyeth W Wasserman, Stephen W Scherer, Jan H Veldink, David R Bentley, Ryan K C Yuen, Melanie Bahlo, and Michael A Eberle. ExpansionHunter denovo: a computational method for locating known and novel repeat expansions in short-read sequencing data. *Genome Biol.*, 21(1):102, April 2020.

[43] Huilong Du, Ying Yu, Yanfei Ma, Qiang Gao, Yinghao Cao, Zhuo Chen, Bin Ma, Ming Qi, Yan Li, Xianfeng Zhao, Jing Wang, Kunfan Liu, Peng Qin, Xin Yang, Lihuang Zhu, Shigui Li, and Chengzhi Liang. Sequencing and de novo assembly of a near complete indica rice genome. *Nat. Commun.*, 8:15324, May 2017.

[44] Richard Durbin, Sean R Eddy, Anders Krogh, and Graeme Mitchison. *Biological*

sequence analysis: probabilistic models of proteins and nucleic acids. Cambridge university press, 1998.

[45] Peter Ebert, Peter A Audano, Qihui Zhu, Bernardo Rodriguez-Martin, David Porubsky, Marc Jan Bonder, Arvis Sulovari, Jana Ebler, Weichen Zhou, Rebecca Serra Mari, Feyza Yilmaz, Xuefang Zhao, Pinghsun Hsieh, Joyce Lee, Sushant Kumar, Jiadong Lin, Tobias Rausch, Yu Chen, Jingwen Ren, Martin Santamarina, Wolfram Höps, Hufsah Ashraf, Nelson T Chuang, Xiaofei Yang, Katherine M Munson, Alexandra P Lewis, Susan Fairley, Luke J Tallon, Wayne E Clarke, Anna O Basile, Marta Byrska-Bishop, André Corvelo, Uday S Evani, Tsung-Yu Lu, Mark J P Chaisson, Junjie Chen, Chong Li, Harrison Brand, Aaron M Wenger, Maryam Ghareghani, William T Harvey, Benjamin Raeder, Patrick Hasenfeld, Allison A Regier, Haley J Abel, Ira M Hall, Paul Flicek, Oliver Stegle, Mark B Gerstein, Jose M C Tubio, Zepeng Mu, Yang I Li, Xinghua Shi, Alex R Hastie, Kai Ye, Zechen Chong, Ashley D Sanders, Michael C Zody, Michael E Talkowski, Ryan E Mills, Scott E Devine, Charles Lee, Jan O Korbel, Tobias Marschall, and Evan E Eichler. Haplotype-resolved diverse human genomes and integrated analysis of structural variation. *Science*, 372(6537), April 2021.

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

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

[48] 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.

[49] 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.

[50] 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.

[51] Gökcen Eraslan, Žiga Avsec, Julien Gagneur, and Fabian J Theis. Deep learning: new computational modelling techniques for genomics. *Nature Reviews Genetics*, 20(7):389–403, 2019.

[52] 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.

[53] 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.

[54] Lauge Farnaes, Amber Hildreth, Nathaly M Sweeney, Michelle M Clark, Shimul Chowdhury, Shareef Nahas, Julie A Cakici, Wendy Benson, Robert H Kaplan, Richard Kronick, and Others. Rapid whole-genome sequencing decreases infant morbidity and cost of hospitalization. *NPJ genomic medicine*, 3(1):1–8, 2018.

[55] 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)—simultaneous clade and personal genome annotation. *Genome Research*, 28(7):1029–1038, 2018.

[56] 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.

[57] Yuriy Fofanov, Yi Luo, Charles Katili, Jim Wang, Yuri Belosludtsev, Thomas Powdrill, Chetan Belapurkar, Viacheslav Fofanov, Tong-Bin Li, Sergey Chumakov, and B Montgomery Pettitt. How independent are the appearances of n-mers in different genomes? *Bioinformatics*, 20(15):2421–2428, October 2004.

[58] G Formenti, A Rhie, B P Walenz, F Thibaud-Nissen, S Koren, E Myers, E D Jarvis, and A M Phillippy. Merfin: improved variant filtering and polishing via k-mer validation. *bioRxiv (to appear)*, 2021.

[59] 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.

[60] 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.

[61] Shuhua Fu, Anqi Wang, and Kin Fai Au. A comparative evaluation of hybrid error correction methods for error-prone long reads. *Genome Biol.*, 20(1):26, February 2019.

[62] Hasindu Gamaarachchi, Hiruna Samarakoon, Sasha P Jenner, James M Ferguson, Timothy G Amos, Jillian M Hammond, Hassaan Saadat, Martin A Smith, Sri Parameswaran, and Ira W Deveson. Fast nanopore sequencing data analysis with slow5. *Nature biotechnology*, pages 1–4, 2022.

[63] 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.

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

[65] Baylor Genetics. Whole genome sequencing test. https://www.baylorgenetics.com/whole-genome-sequencing/, November 2019. Accessed: 2021-5-23.

[66] Rady Genomics. Services offered. https://radygenomics.org/clinical-genome-services/offerings/, March 2021. Accessed: 2021-5-23.

[67] Ariel Gershman, Michael E G Sauria, Paul W Hook, Savannah J Hoyt, Roham Razaghi, Sergey Koren, Nicolas Altemose, Gina V Caldas, Mitchell R Vollger, Glennis A Logsdon, Arang Rhie, Evan E Eichler, Michael C Schatz, Rachel J O'Neill, Adam M Phillippy, Karen H Miga, and Winston Timp. Epigenetic patterns in a complete human genome. *bioRxiv*, page 2021.05.26.443420, May 2021.

[68] Ariel Gershman, Michael EG Sauria, Paul W Hook, Savannah J Hoyt, Roham Razaghi, Sergey Koren, Nicolas Altemose, Gina V Caldas, Mitchell R Vollger, Glennis A Logsdon, et al. Epigenetic patterns in a complete human genome. *bioRxiv*, 2021.

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

[70] Rachel L Goldfeder and Euan A Ashley. A precision metric for clinical genome sequencing. *bioRxiv*, page 051490, January 2016.

[71] Todd R Golub, Donna K Slonim, Pablo Tamayo, Christine Huard, Michelle Gaasenbeek, Jill P Mesirov, Hilary Coller, Mignon L Loh, James R Downing, Mark A Caligiuri, et al. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *science*, 286(5439):531–537, 1999.

[72] Nina Gonzaludo, John W Belmont, Vladimir G Gainullin, and Ryan J Taft. Estimating the burden and economic impact of pediatric genetic disease. *Genet. Med.*, 21(8):1781–1789, August 2019.

[73] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[74] David Gordon, John Huddleston, Mark JP Chaisson, Christopher M Hill, Zev N Kronenberg, Katherine M Munson, Maika Malig, Archana Raja, Ian Fiddes, LaDeana W Hillier, et al. Long-read sequence assembly of the gorilla genome. *Science*, 352(6281), 2016.

[75] 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.

[76] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376, 2006.

[77] Wilfried M Guiblet, Marzia A Cremona, Monika Cechova, Robert S Harris, Iva Kejnovská, Eduard Kejnovsky, Kristin Eckert, Francesca Chiaromonte, and Kateryna D Makova. Long-read sequencing technology indicates genome-wide effects of non-b DNA on polymerization speed and error rate. *Genome Res.*, 28(12):1767–1778, December 2018.

[78] Andrew P Han. Human pangenome reference consortium releases data from 30 genomes. *genomeweb.com*, March 2021. [Online. Retrieved June 14, 2021.].

[79] 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.

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

[81] Philip Hieter and Mark Boguski. Functional genomics: it's all how you read it. *Science*, 278(5338):601–602, 1997.

[82] 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.

[83] Matthew B Hufford, Arun S Seetharam, Margaret R Woodhouse, Kapeel M Chougule, Shujun Ou, Jianing Liu, William A Ricci, Tingting Guo, Andrew Olson, Yinjie Qiu, Rafael Della Coletta, Silas Tittes, Asher I Hudson, Alexandre P Marand, Sharon Wei, Zhenyuan Lu, Bo Wang, Marcela K Tello-Ruiz, Rebecca D Piri, Na Wang, Dong Won Kim, Yibing Zeng, Christine H O'Connor, Xianran Li, Amanda M Gilbert, Erin Baggs, Ksenia V Krasileva, John L Portwood, Ethalinda K S Cannon, Carson M Andorf, Nancy Manchanda, Samantha J Snodgrass, David E Hufnagel, Qiuhan Jiang, Sarah Pedersen, Michael L Syring, David A Kudrna, Victor Llaca, Kevin Fengler, Robert J Schmitz, Jeffrey Ross-Ibarra, Jianming Yu, Jonathan I Gent, Candice N Hirsch, Doreen Ware, and R Kelly Dawe. De novo assembly, annotation, and comparative analysis of 26 diverse maize genomes. *bioRxiv*, page 2021.01.14.426684, January 2021.

[84] C Jain, A Rhie, N Hansen, S Koren, and A M Phillippy. A long read mapping method for highly repetitive reference sequences. *bioRxiv*, 2020.

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

[86] Chirag Jain, Arang Rhie, Haowen Zhang, Claudia Chu, Brian P Walenz, Sergey Koren, and Adam M Phillippy. Weighted minimizer sampling improves long read mapping. *Bioinformatics*, 36(Suppl_1):i111–i118, July 2020.

[87] 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.

[88] Miten Jain, Sergey Koren, Karen H Miga, Josh Quick, Arthur C Rand, Thomas A Sasani, John R Tyson, Andrew D Beggs, Alexander T Dilthey, Ian T Fiddes, et al. Nanopore sequencing and assembly of a human genome with ultra-long reads. *Nature biotechnology*, 36(4):338, 2018.

[89] Miten Jain, Sergey Koren, Karen H Miga, Josh Quick, Arthur C Rand, Thomas A Sasani, John R Tyson, Andrew D Beggs, Alexander T Dilthey, Ian T Fiddes, Sunir Malla, Hannah Marriott, Tom Nieto, Justin O'Grady, Hugh E Olsen, Brent S Pedersen, Arang Rhie, Hollian Richardson, Aaron R Quinlan, Terrance P Snutch, Louise Tee, Benedict Paten, Adam M Phillippy, Jared T Simpson, Nicholas J

Loman, and Matthew Loose. Nanopore sequencing and assembly of a human genome with ultra-long reads. *Nat. Biotechnol.*, 36(4):338–345, April 2018.

[90] 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.

[91] Wenzel Jakob, Jason Rhinelander, and Dean Moldovan. pybind11—seamless operability between c++ 11 and python, 2016.

[92] 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.

[93] 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.

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

[95] 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.

[96] 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.

[97] 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.

[98] 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), 2018.

[99] Peter Krusche, Len Trigg, Paul C Boutros, Christopher E Mason, M Francisco, Benjamin L Moore, Mar Gonzalez-Porta, Michael A Eberle, Zivana Tezak, Samir

Lababidi, et al. Best practices for benchmarking germline small-variant calls in human genomes. *Nature biotechnology*, 37(5):555–560, 2019.

[100] Stefan Kurtz, Adam Phillippy, Arthur L Delcher, Michael Smoot, Martin Shumway, Corina Antonescu, and Steven L Salzberg. Versatile and open software for comparing large genomes. *Genome biology*, 5(2):R12, 2004.

[101] Avantika Lal, Michael Brown, Rahul Mohan, Joyjit Daw, James Drake, and Johnny Israeli. Improving long-read consensus sequencing accuracy with deep learning. *bioRxiv*, 2021.

[102] Dandan Lang, Shilai Zhang, Pingping Ren, Fan Liang, Zongyi Sun, Guanliang Meng, Yuntao Tan, Xiaokang Li, Qihua Lai, Lingling Han, Depeng Wang, Fengyi Hu, Wen Wang, and Shanlin Liu. Comparison of the two up-to-date sequencing technologies for genome assembly: HiFi reads of pacific biosciences sequel II system and ultralong reads of oxford nanopore, 2020.

[103] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

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

[105] Hane Lee, Joshua L Deignan, Naghmeh Dorrani, Samuel P Strom, Sibel Kantarci, Fabiola Quintero-Rivera, Kingshuk Das, Traci Toy, Bret Harry, Michael Yourshaw, Michelle Fox, Brent L Fogel, Julian A Martinez-Agosto, Derek A Wong, Vivian Y Chang, Perry B Shieh, Christina G S Palmer, Katrina M Dipple, Wayne W Grody, Eric Vilain, and Stanley F Nelson. Clinical exome sequencing for genetic identification of rare mendelian disorders. *JAMA*, 312(18):1880–1887, November 2014.

[106] 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.

[107] 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.

[108] 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.

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

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

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

[112] 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.

[113] Heng Li, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, and Richard Durbin. The sequence alignment/map format and samtools. *Bioinformatics*, 25(16):2078–2079, 2009.

[114] Heng Li, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, Richard Durbin, and 1000 Genome Project Data Processing Subgroup. The sequence Alignment/Map format and SAMtools. *Bioinformatics*, 25(16):2078–2079, August 2009.

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

[116] Jianing Liu, Arun S Seetharam, Kapeel Chougule, Shujun Ou, Kyle W Swentowsky, Jonathan I Gent, Victor Llaca, Margaret R Woodhouse, Nancy Manchanda, Gernot G Presting, David A Kudrna, Magdy Alabady, Candice N Hirsch, Kevin A Fengler, Doreen Ware, Todd P Michael, Matthew B Hufford, and R Kelly Dawe. Gapless assembly of maize chromosomes using long-read technologies. *Genome Biol.*, 21(1):121, May 2020.

[117] Qian Liu, Li Fang, Guoliang Yu, Depeng Wang, Chuan-Le Xiao, and Kai Wang. Detection of dna base modifications by deep recurrent neural network on oxford nanopore sequencing data. *Nature communications*, 10(1):1–11, 2019.

[118] Glennis A Logsdon, Mitchell R Vollger, and Evan E Eichler. Long-read human genome sequencing and its applications. *Nat. Rev. Genet.*, 21(10):597–614, October 2020.

[119] Glennis A Logsdon, Mitchell R Vollger, and Evan E Eichler. Long-read human genome sequencing and its applications. *Nature Reviews Genetics*, 21(10):597–614, 2020.

[120] 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.

[121] 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.

[122] Nicholas J Loman, Joshua Quick, and Jared T Simpson. A complete bacterial genome assembled de novo using only nanopore sequencing data. *Nat. Methods*, 12(8):733–735, August 2015.

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

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

[125] 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.

[126] 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.

[127] Medhat Mahmoud, Nastassia Gobet, Diana Ivette Cruz-Dávalos, Ninon Mounier, Christophe Dessimoz, and Fritz J Sedlazeck. Structural variant calling: the long and the short of it. *Genome Biol.*, 20(1):246, November 2019.

[128] Ann M Mc Cartney, Kishwar Shafin, Michael Alonge, Andrey V Bzikadze, Giulio Formenti, Arkarachai Fungtammasan, Kerstin Howe, Chirag Jain, Sergey Koren, Glennis A Logsdon, et al. Chasing perfection: validation and polishing strategies for telomere-to-telomere genome assemblies. *biorxiv*, 2021.

[129] Shawn E McCandless, Jeanne W Brunger, and Suzanne B Cassidy. The burden

of genetic disease on inpatient care in a children's hospital. *Am. J. Hum. Genet.*, 74(1):121–127, January 2004.

[130] 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.

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

[132] Larry R Medsker and LC Jain. Recurrent neural networks. *Design and Applications*, 5:64–67, 2001.

[133] Jason D Merker, Aaron M Wenger, Tam Sneddon, Megan Grove, Zachary Zappala, Laure Fresard, Daryl Waggott, Sowmi Utiramerur, Yanli Hou, Kevin S Smith, Stephen B Montgomery, Matthew Wheeler, Jillian G Buchan, Christine C Lambert, Kevin S Eng, Luke Hickey, Jonas Korlach, James Ford, and Euan A Ashley. Long-read genome sequencing identifies causal structural variation in a mendelian disease. *Genet. Med.*, 20(1):159–163, January 2018.

[134] Michael L Metzker. Sequencing technologies - the next generation. *Nat. Rev. Genet.*, 11(1):31–46, January 2010.

[135] 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.

[136] Karen H Miga, Sergey Koren, Arang Rhie, Mitchell R Vollger, Ariel Gershman, Andrey Bzikadze, Shelise Brooks, Edmund Howe, David Porubsky, Glennis A Logsdon, Valerie A Schneider, Tamara Potapova, Jonathan Wood, William Chow, Joel Armstrong, Jeanne Fredrickson, Evgenia Pak, Kristof Tigyi, Milinn Kremitzki, Christopher Markovic, Valerie Maduro, Amalia Dutra, Gerard G Bouffard, Alexander M Chang, Nancy F Hansen, Amy B Wilfert, Françoise Thibaud-Nissen, Anthony D Schmitt, Jon-Matthew Belton, Siddarth Selvaraj, Megan Y Dennis, Daniela C Soto, Ruta Sahasrabudhe, Gulhan Kaya, Josh Quick, Nicholas J Loman, Nadine Holmes, Matthew Loose, Urvashi Surti, Rosa Ana Risques, Tina A Graves Lindsay, Robert Fulton, Ira Hall, Benedict Paten, Kerstin Howe, Winston Timp, Alice Young, James C Mullikin, Pavel A Pevzner, Jennifer L Gerton, Beth A Sullivan, Evan E Eichler, and Adam M Phillippy. Telomere-to-telomere assembly of a complete human X chromosome. *Nature*, 585(7823):79–84, September 2020.

[137] Alla Mikheenko, Andrey V Bzikadze, Alexey Gurevich, Karen H Miga, and Pavel A Pevzner. TandemTools: mapping long reads and assessing/improving assembly quality in extra-long tandem repeats. *Bioinformatics*, 36(Suppl_1):i75–i83, July 2020.

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

[139] Danny E Miller, Arvis Sulovari, Tianyun Wang, Hailey Loucks, Kendra Hoekzema, Katherine M Munson, Alexandra P Lewis, Edith P Almanza Fuerte, Catherine R Paschal, Jenny Thies, James T Bennett, Ian Glass, Katrina M Dipple, Karynne Patterson, Emily S Bonkowski, Zoe Nelson, Audrey Squire, Megan Sikes, Erika Beckman, Robin L Bennett, Dawn Earl, Winston Lee, Rando Allikmets, Seth J Perlman, Penny Chow, Anne V Hing, Margaret P Adam, Angela Sun, Christina Lam, Irene Chang, University of Washington Center for Mendelian Genomics, Tim Cherry, Jessica X Chong, Michael J Bamshad, Deborah A Nickerson, Heather C Mefford, Dan Doherty, and Evan E Eichler. Targeted long-read sequencing resolves complex structural variants and identifies missing disease-causing variants. *bioRxiv*, page 2020.11.03.365395, November 2020.

[140] 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.

[141] Seonwoo Min, Byunghan Lee, and Sungroh Yoon. Deep learning in bioinformatics. *Briefings in bioinformatics*, 18(5):851–869, 2017.

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

[143] M Naish, M Alonge, P Wlodzimierz, A J Tock, and others. The genetic and epigenetic landscape of the arabidopsis centromeres. *bioRxiv*, 2021.

[144] Maria Nattestad and Calvin Bao. GitHub - dnanexus/dot: Dot: An interactive dot plot viewer for comparative genomics.

[145] Whitney K Newey. Adaptive estimation of regression models via moment restrictions. *Journal of Econometrics*, 38(3):301–339, 1988.

[146] Peng Ni, Neng Huang, Zhi Zhang, De-Peng Wang, Fan Liang, Yu Miao, Chuan-Le Xiao, Feng Luo, and Jianxin Wang. Deepsignal: detecting dna methylation state

from nanopore sequencing reads using deep-learning. *Bioinformatics*, 35(22):4586–4595, 2019.

[147] Sergey Nurk, Sergey Koren, Arang Rhie, Mikko Rautiainen, Andrey V Bzikadze, Alla Mikheenko, Mitchell R Vollger, Nicolas Altemose, Lev Uralsky, Ariel Gershman, Sergey Aganezov, Savannah J Hoyt, Mark Diekhans, Glennis A Logsdon, Michael Alonge, Stylianos E Antonarakis, Matthew Borchers, Gerard G Bouffard, Shelise Y Brooks, Gina V Caldas, Haoyu Cheng, Chen-Shan Chin, William Chow, Leonardo G de Lima, Philip C Dishuck, Richard Durbin, Tatiana Dvorkina, Ian T Fiddes, Giulio Formenti, Robert S Fulton, Arkarachai Fungtammasan, Erik Garrison, Patrick G S Grady, Tina A Graves-Lindsay, Ira M Hall, Nancy F Hansen, Gabrielle A Hartley, Marina Haukness, Kerstin Howe, Michael W Hunkapiller, Chirag Jain, Miten Jain, Erich D Jarvis, Peter Kerpedjiev, Melanie Kirsche, Mikhail Kolmogorov, Jonas Korlach, Milinn Kremitzki, Heng Li, Valerie V Maduro, Tobias Marschall, Ann M McCartney, Jennifer McDaniel, Danny E Miller, James C Mullikin, Eugene W Myers, Nathan D Olson, Benedict Paten, Paul Peluso, Pavel A Pevzner, David Porubsky, Tamara Potapova, Evgeny I Rogaev, Jeffrey A Rosenfeld, Steven L Salzberg, Valerie A Schneider, Fritz J Sedlazeck, Kishwar Shafin, Colin J Shew, Alaina Shumate, Yumi Sims, Arian F A Smit, Daniela C Soto, Ivan Sović, Jessica M Storer, Aaron Streets, Beth A Sullivan, Françoise Thibaud-Nissen, James Torrance, Justin Wagner, Brian P Walenz, Aaron Wenger, Jonathan M D Wood, Chunlin Xiao, Stephanie M Yan, Alice C Young, Samantha Zarate, Urvashi Surti, Rajiv C McCoy, Megan Y Dennis, Ivan A Alexandrov, Jennifer L Gerton, Rachel J O'Neill, Winston Timp, Justin M Zook, Michael C Schatz, Evan E Eichler, Karen H Miga, and Adam M Phillippy. The complete sequence of a human genome. *bioRxiv*, page 2021.05.26.445798, May 2021.

[148] 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.

[149] 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.

[150] 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 Res.*, 30(9):1291–1305, September 2020.

[151] NVIDIA. NVIDIA clara parabricks. https://www.nvidia.com/en-us/clara/

genomics/. Accessed: 2021-5-25.

[152] 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.

[153] Nathan D Olson, Justin Wagner, Jennifer McDaniel, Sarah H Stephens, Samuel T Westreich, Anish G Prasanna, Elaine Johanson, Emily Boja, Ezekiel J Maier, Omar Serang, et al. precisionfda truth challenge v2: Calling variants from short-and long-reads in difficult-to-map regions. *bioRxiv*, 2020.

[154] Nathan D Olson, Justin Wagner, Jennifer McDaniel, Sarah H Stephens, Samuel T Westreich, Anish G Prasanna, Elaine Johanson, Emily Boja, Ezekiel J Maier, Omar Serang, David Jáspez, José M Lorenzo-Salazar, Adrián Muñoz-Barrera, Luis A Rubio-Rodríguez, Carlos Flores, Konstantinos Kyriakidis, Andigoni Malousi, Kishwar Shafin, Trevor Pesout, Miten Jain, Benedict Paten, Pi-Chuan Chang, Alexey Kolesnikov, Maria Nattestad, Gunjan Baid, Sidharth Goel, Howard Yang, Andrew Carroll, Robert Eveleigh, Mathieu Bourgey, Guillaume Bourque, Gen Li, M A ChouXian, Linqi Tang, D U YuanPing, Shaowei Zhang, Jordi Morata, Raúl Tonda, Genís Parra, Jean-Rémi Trotta, Christian Brueffer, Sinem Demirkaya-Budak, Duygu Kabakci-Zorlu, Deniz Turgut, Özem Kalay, Gungor Budak, Kübra Narcı, Elif Arslan, Richard Brown, Ivan J Johnson, Alexey Dolgoborodov, Vladimir Semenyuk, Amit Jain, H Serhat Tetikol, Varun Jain, Mike Ruehle, Bryan Lajoie, Cooper Roddey, Severine Catreux, Rami Mehio, Mian Umair Ahsan, Qian Liu, Kai Wang, Sayed Mohammad Ebrahim Sahraeian, Li Tai Fang, Marghoob Mohiyuddin, Calvin Hung, Chirag Jain, Hanying Feng, Zhipan Li, Luoqi Chen, Fritz J Sedlazeck, and Justin M Zook. precisionFDA truth challenge v2: Calling variants from short- and long-reads in difficult-to-map regions. *bioRxiv*, page 2020.11.13.380741, February 2021.

[155] Mallory J Owen, Anna-Kaisa Niemi, David P Dimmock, Mark Speziale, Mark Nespeca, Kevin K Chau, Luca Van Der Kraan, Meredith S Wright, Christian Hansen, Narayanan Veeraraghavan, Yan Ding, Jerica Lenberg, Shimul Chowdhury, Charlotte A Hobbs, Sergey Batalov, Zhanyang Zhu, Shareef A Nahas, Sheldon Gilmer, Gail Knight, Sebastien Lefebvre, John Reynders, Thomas Defay, Jacqueline Weir, Vicki S Thomson, Louise Fraser, Bryan R Lajoie, Tim K McPhail, Shyamal S Mehtalia, Chris M Kunard, Kevin P Hall, and Stephen F Kingsmore. Rapid Sequencing-Based diagnosis of thiamine metabolism dysfunction syndrome. *N. Engl. J. Med.*, 384(22):2159–2161, June 2021.

[156] Oxford Nanopore Technologies. https://github.com/nanoporetech/medaka. https://github.com/nanoporetech/medaka. Accessed: 2021-2-5.

[157] pacbio. Data release: Highest-quality, most contiguous individual human genome assembly to date.

[158] PacBio. GenomicConsensus.

[159] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017.

[160] 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.

[161] 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.

[162] Benedict Paten, Adam M Novak, Jordan M Eizenga, and Erik Garrison. Genome graphs and the evolution of genome inference. *Genome research*, 27(5):665–676, 2017.

[163] 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.

[164] Alexander Payne, Nadine Holmes, Thomas Clarke, Rory Munro, Bisrat J Debebe, and Matthew Loose. Readfish enables targeted nanopore sequencing of gigabase-sized genomes. *Nature biotechnology*, 39(4):442–450, 2021.

[165] Pavel A Pevzner, Haixu Tang, and Michael S Waterman. An Eulerian path approach to DNA fragment assembly. *Proceedings of the National Academy of Sciences*, 98(17):9748–9753, 2001.

[166] Ryan Poplin, Pi-Chuan Chang, David Alexander, Scott Schwartz, Thomas Colthurst, Alexander Ku, Dan Newburger, Jojo Dijamco, Nam Nguyen, Pegah T Afshar, et al. A universal snp and small-indel variant caller using deep neural networks. *Nature biotechnology*, 36(10):983, 2018.

[167] Ryan Poplin, Pi-Chuan Chang, David Alexander, Scott Schwartz, Thomas Colthurst, Alexander Ku, Dan Newburger, Jojo Dijamco, Nam Nguyen, Pegah T Afshar, Sam S Gross, Lizzie Dorfman, Cory Y McLean, and Mark A DePristo.

A universal SNP and small-indel variant caller using deep neural networks. *Nat. Biotechnol.*, 36(10):983–987, November 2018.

[168] 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.

[169] Ploy N Pratanwanich, Fei Yao, Ying Chen, Casslynn WQ Koh, Yuk Kei Wan, Christopher Hendra, Polly Poon, Yeek Teck Goh, Phoebe ML Yap, Jing Yuan Chooi, et al. Identification of differential rna modifications from nanopore direct rna sequencing with xpore. *Nature Biotechnology*, 39(11):1394–1402, 2021.

[170] James R Priest, Scott R Ceresnak, Frederick E Dewey, Lindsey E Malloy-Walton, Kyla Dunn, Megan E Grove, Marco V Perez, Katsuhide Maeda, Anne M Dubin, and Euan A Ashley. Molecular diagnosis of long QT syndrome at 10 days of life by rapid whole genome sequencing. *Heart Rhythm*, 11(10):1707–1713, October 2014.

[171] James Rush Priest, Charles Gawad, Kristopher M Kahlig, Joseph K Yu, Thomas O'Hara, Patrick M Boyle, Sridharan Rajamani, Michael J Clark, Sarah T K Garcia, Scott Ceresnak, Jason Harris, Sean Boyle, Frederick E Dewey, Lindsey Malloy-Walton, Kyla Dunn, Megan Grove, Marco V Perez, Norma F Neff, Richard Chen, Katsuhide Maeda, Anne Dubin, Luiz Belardinelli, John West, Christian Antolik, Daniela Macaya, Thomas Quertermous, Natalia A Trayanova, Stephen R Quake, and Euan A Ashley. Early somatic mosaicism is a rare cause of long-QT syndrome. *Proc. Natl. Acad. Sci. U. S. A.*, 113(41):11555–11560, October 2016.

[172] 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.

[173] Arthur C Rand, Miten Jain, Jordan M Eizenga, Audrey Musselman-Brown, Hugh E Olsen, Mark Akeson, and Benedict Paten. Mapping dna methylation with high-throughput nanopore sequencing. *Nature methods*, 14(4):411–413, 2017.

[174] 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.

[175] Mikko Rautiainen and Tobias Marschall. Mbg: Minimizer-based sparse de bruijn graph construction. *Bioinformatics*, 37(16):2476–2478, 2021.

[176] Arang Rhie, Shane A McCarthy, Olivier Fedrigo, Joana Damas, Giulio Formenti, Sergey Koren, Marcela Uliano-Silva, William Chow, Arkarachai Fungtammasan, Gregory L Gedman, Lindsey J Cantin, Francoise Thibaud-Nissen, Leanne Haggerty, Chul Lee, Byung June Ko, Juwan Kim, Iliana Bista, Michelle Smith, Bettina Haase, Jacquelyn Mountcastle, Sylke Winkler, Sadye Paez, Jason Howard, Sonja C Vernes, Tanya M Lama, Frank Grutzner, Wesley C Warren, Christopher Balakrishnan, Dave Burt, Julia M George, Mathew Biegler, David Iorns, Andrew Digby, Daryl Eason, Taylor Edwards, Mark Wilkinson, George Turner, Axel Meyer, Andreas F Kautt, Paolo Franchini, H William Detrich, Hannes Svardal, Maximilian Wagner, Gavin J P Naylor, Martin Pippel, Milan Malinsky, Mark Mooney, Maria Simbirsky, Brett T Hannigan, Trevor Pesout, Marlys Houck, Ann Misuraca, Sarah B Kingan, Richard Hall, Zev Kronenberg, Jonas Korlach, Ivan Sović, Christopher Dunn, Zemin Ning, Alex Hastie, Joyce Lee, Siddarth Selvaraj, Richard E Green, Nicholas H Putnam, Jay Ghurye, Erik Garrison, Ying Sims, Joanna Collins, Sarah Pelan, James Torrance, Alan Tracey, Jonathan Wood, Dengfeng Guan, Sarah E London, David F Clayton, Claudio V Mello, Samantha R Friedrich, Peter V Lovell, Ekaterina Osipova, Farooq O Al-Ajli, Simona Secomandi, Heebal Kim, Constantina Theofanopoulou, Yang Zhou, Robert S Harris, Kateryna D Makova, Paul Medvedev, Jinna Hoffman, Patrick Masterson, Karen Clark, Fergal Martin, Kevin Howe, Paul Flicek, Brian P Walenz, Woori Kwak, Hiram Clawson, Mark Diekhans, Luis Nassar, Benedict Paten, Robert H S Kraus, Harris Lewin, Andrew J Crawford, M Thomas P Gilbert, Guojie Zhang, Byrappa Venkatesh, Robert W Murphy, Klaus-Peter Koepfli, Beth Shapiro, Warren E Johnson, Federica Di Palma, Tomas Margues-Bonet, Emma C Teeling, Tandy Warnow, Jennifer Marshall Graves, Oliver A Ryder, David Hausler, Stephen J O'Brien, Kerstin Howe, Eugene W Myers, Richard Durbin, Adam M Phillippy, and Erich D Jarvis. Towards complete and error-free genome assemblies of all vertebrate species. *Cold Spring Harbor Laboratory*, page 2020.05.22.110833, May 2020.

[177] Arang Rhie, Brian P Walenz, Sergey Koren, and Adam M Phillippy. Merqury: reference-free quality, completeness, and phasing assessment for genome assemblies. *Genome Biol.*, 21(1):245, September 2020.

[178] Anthony Rhoads and Kin Fai Au. Pacbio sequencing and its applications. *Genomics, proteomics & bioinformatics*, 13(5):278–289, 2015.

[179] Joseph W F Robertson, Madhav L Ghimire, and Joseph E Reiner. Nanopore sensing: A physical-chemical approach. *Biochim. Biophys. Acta Biomembr.*, 1863(9):183644, May 2021.

[180] 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.

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

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

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

[184] Steven L Salzberg, Adam M Phillippy, Aleksey Zimin, Daniela Puiu, Tanja Magoc, Sergey Koren, Todd J Treangen, Michael C Schatz, Arthur L Delcher, Michael Roberts, Guillaume Marçais, Mihai Pop, and James A Yorke. GAGE: A critical evaluation of genome assemblies and assembly algorithms. *Genome Res.*, 22(3):557–567, March 2012.

[185] 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.

[186] Fritz J Sedlazeck, Zachary Lemmon, Sebastian Soyk, William J Salerno, Zachary Lippman, and Michael C Schatz. Svcollector: Optimized sample selection for validating and long-read resequencing of structural variants. *BioRxiv*, page 342386, 2018.

[187] Fritz J Sedlazeck, Philipp Rescheneder, Moritz Smolka, Han Fang, Maria Nattestad, Arndt von Haeseler, and Michael C Schatz. Accurate detection of complex structural variations using single-molecule sequencing. *Nat Methods*, 15(6):461–468, 2018.

[188] Fritz J Sedlazeck, Philipp Rescheneder, Moritz Smolka, Han Fang, Maria Nattestad, Arndt von Haeseler, and Michael C Schatz. Accurate detection of complex structural variations using single-molecule sequencing. *Nat. Methods*, 15(6):461–468, June 2018.

[189] Fritz J Sedlazeck, Philipp Rescheneder, Moritz Smolka, Han Fang, Maria Nattestad, Arndt von Haeseler, and Michael C Schatz. Accurate detection of complex structural variations using single-molecule sequencing. *Nat. Methods*, 15(6):461–468, April 2018.

[190] Mantas Sereika, Rasmus Hansen Kirkegaard, Søren Michael Karst, Thomas Yssing Michaelsen, Emil Aarre Sørensen, Rasmus Dam Wollenberg, and Mads Albertsen. Oxford nanopore r10. 4 long-read sequencing enables near-perfect bacterial genomes

from pure cultures and metagenomes without short-read or reference polishing. *bioRxiv*, 2021.

[191] Kishwar Shafin, Trevor Pesout, Pi-Chuan Chang, Maria Nattestad, Alexey Kolesnikov, Sidharth Goel, Gunjan Baid, Jordan M Eizenga, Karen H Miga, Paolo Carnevali, Miten Jain, Andrew Carroll, and Benedict Paten. Haplotype-aware variant calling enables high accuracy in nanopore long-reads using deep neural networks. *bioRxiv*, page 2021.03.04.433952, March 2021.

[192] Kishwar Shafin, Trevor Pesout, Pi-Chuan Chang, Maria Nattestad, Alexey Kolesnikov, Sidharth Goel, Gunjan Baid, Jordan M Eizenga, Karen H Miga, Paolo Carnevali, Miten Jain, Andrew Carroll, and Benedict Paten. Haplotype-aware variant calling enables high accuracy in nanopore long-reads using deep neural networks. *bioRxiv*, page 2021.03.04.433952, March 2021.

[193] 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*, 18(11):1322–1332, 2021.

[194] Kishwar Shafin, Trevor Pesout, Ryan Lorig-Roach, Marina Haukness, Hugh E Olsen, Colleen Bosworth, Joel Armstrong, Kristof Tigyi, Nicholas Maurer, Sergey Koren, et al. Nanopore sequencing and the shasta toolkit enable efficient de novo assembly of eleven human genomes. *Nature Biotechnology*, pages 1–10, 2020.

[195] Jared T Simpson, Kim Wong, Shaun D Jackman, Jacqueline E Schein, Steven J M Jones, and Inanç Birol. ABySS: a parallel assembler for short read sequence data. *Genome Res.*, 19(6):1117–1123, June 2009.

[196] Jared T Simpson, Rachael E Workman, PC Zuzarte, Matei David, LJ Dursi, and Winston Timp. Detecting dna cytosine methylation using nanopore sequencing. *Nature methods*, 14(4):407–410, 2017.

[197] Felipe A. Simão, 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.

[198] Jouni Sirén, Jean Monlong, Xian Chang, Adam M Novak, Jordan M Eizenga, Charles Markello, Jonas A Sibbesen, Glenn Hickey, Pi-Chuan Chang, Andrew Carroll, et al. Pangenomics enables genotyping of known structural variants in 5202 diverse genomes. *Science*, 374(6574):abg8871, 2021.

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

[200] Sneha D. Goenka*, John E. Gorzynski*, Kishwar Shafin*, Dianna G. Fisk, Trevor Pesout , Jean Monlong , Tanner D. Jensen , Pi-Chuan Chang, Gunjan Baid, Jonathan A Bernstein, Jeffrey Christle , Karen P. Dalton, Daniel R. Garalde, Megan E. Grove, Joseph Guillory, Alexey Kolesnikov, Maria Nattestad, Maura R.Z. Ruzhnikov, Mehrzad Samadi, Ankit Sethia, Elizabeth Spiteri, Chris Wright, Katherine Xiong, Tong Zhu, Miten Jain, Fritz J. Sedlazeck, Andrew Carroll, Benedict Paten, Euan A. Ashley. Technical development of rapid whole genome nanopore sequencing and variant identification pipeline. *Unpublished*, 2021.

[201] Kimberly Splinter, David R Adams, Carlos A Bacino, Hugo J Bellen, Jonathan A Bernstein, Alys M Cheatle-Jarvela, Christine M Eng, Cecilia Esteves, William A Gahl, Rizwan Hamid, Howard J Jacob, Bijal Kikani, David M Koeller, Isaac S Kohane, Brendan H Lee, Joseph Loscalzo, Xi Luo, Alexa T McCray, Thomas O Metz, John J Mulvihill, Stanley F Nelson, Christina G S Palmer, John A Phillips, 3rd, Leslie Pick, John H Postlethwait, Chloe Reuter, Vandana Shashi, David A Sweetser, Cynthia J Tifft, Nicole M Walley, Michael F Wangler, Monte Westerfield, Matthew T Wheeler, Anastasia L Wise, Elizabeth A Worthey, Shinya Yamamoto, Euan A Ashley, and Undiagnosed Diseases Network. Effect of genetic diagnosis on patients with previously undiagnosed disease. *N. Engl. J. Med.*, 379(22):2131–2139, November 2018.

[202] 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.

[203] Nathaly M Sweeney, Shareef A Nahas, Sh Chowdhury, Sergey Batalov, Michelle Clark, Sara Caylor, Julie Cakici, John J Nigro, Yan Ding, Narayanan Veeraraghavan, and Others. Rapid whole genome sequencing impacts care and resource utilization in infants with congenital heart disease. *NPJ genomic medicine*, 6(1):1–10, 2021.

[204] 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.

[205] Ryan Tewhey, Vikas Bansal, Ali Torkamani, Eric J Topol, and Nicholas J Schork.

The importance of phase information for human genomics. *Nat. Rev. Genet.*, 12(3):215–223, March 2011.

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

[207] 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.

[208] Erwin L van Dijk, Hélène Auger, Yan Jaszczyszyn, and Claude Thermes. Ten years of next-generation sequencing technology. *Trends Genet.*, 30(9):418–426, September 2014.

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

[210] 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.

[211] Jenny Mai Vo, Logan Mulroney, Jen Quick-Cleveland, Miten Jain, Mark Akeson, and Manuel Ares. Synthesis of modified nucleotide polymers by the poly (u) polymerase cid1: application to direct rna sequencing on nanopores. *RNA*, 27(12):1497–1511, 2021.

[212] M R Vollger, X Guitart, P C Dishuck, L Mercuri, and others. Segmental duplications and their variation in a complete human genome. *bioRxiv*, 2021.

[213] Mitchell R Vollger, Xavi Guitart, Philip C Dishuck, Ludovica Mercuri, William T Harvey, Ariel Gershman, Mark Diekhans, Arvis Sulovari, Katherine M Munson, Alexandra M Lewis, Kendra Hoekzema, David Porubsky, Ruiyang Li, Sergey Nurk, Sergey Koren, Karen H Miga, Adam M Phillippy, Winston Timp, Mario Ventura, and Evan E Eichler. Segmental duplications and their variation in a complete human genome. *bioRxiv*, page 2021.05.26.445678, May 2021.

[214] 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.

[215] 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.

[216] Justin Wagner, Nathan D Olson, Lindsay Harris, Ziad Khan, Jesse Farek, Medhat Mahmoud, Ana Stankovic, Vladimir Kovacevic, Aaron M Wenger, William J Rowell, et al. Benchmarking challenging small variants with linked and long reads. *BioRxiv*, 2020.

[217] Justin Wagner, Nathan D Olson, Lindsay Harris, Jennifer McDaniel, Haoyu Cheng, Arkarachai Fungtammasan, Yih-Chii Hwang, Richa Gupta, Aaron M Wenger, William J Rowell, Ziad M Khan, Jesse Farek, Yiming Zhu, Aishwarya Pisupati, Medhat Mahmoud, Chunlin Xiao, Byunggil Yoo, Sayed Mohammad Ebrahim Sahraeian, Danny E Miller, David Jáspez, José M Lorenzo-Salazar, Adrián Muñoz-Barrera, Luis A Rubio-Rodríguez, Carlos Flores, Giuseppe Narzisi, Uday Shanker Evani, Wayne E Clarke, Joyce Lee, Christopher E Mason, Stephen E Lincoln, Karen H Miga, Mark T W Ebbert, Alaina Shumate, Heng Li, Chen-Shan Chin, Justin M Zook, and Fritz J Sedlazeck. Towards a comprehensive variation benchmark for challenging Medically-Relevant autosomal genes. *bioRxiv*, page 2021.06.07.444885, June 2021.

[218] 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.

[219] Mick Watson. Mind the gaps – ignoring errors in long read assemblies critically affects protein prediction.

[220] 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.

[221] 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.

[222] 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.

[223] 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, Armin Töpfer, Michael Alonge, Medhat Mahmoud, Yufeng Qian, Chen-Shan Chin, Adam M Phillippy, Michael C Schatz, Gene Myers, Mark A DePristo, Jue Ruan, Tobias Marschall, Fritz J Sedlazeck, Justin M Zook, Heng Li, Sergey Koren, Andrew Carroll, David R Rank, and Michael W Hunkapiller. Accurate circular consensus long-read sequencing improves variant detection and assembly of a human genome. *Nat. Biotechnol.*, 37(10):1155–1162, October 2019.

[224] Ryan R Wick, Louise M Judd, and Kathryn E Holt. Comparison of oxford nanopore basecalling tools. *January. https://doi. org*, 10, 2018.

[225] Ryan R Wick, Louise M Judd, and Kathryn E Holt. Performance of neural network basecalling tools for oxford nanopore sequencing. *Genome biology*, 20(1):1–10, 2019.

[226] Yaping Yang, Donna M Muzny, Jeffrey G Reid, Matthew N Bainbridge, Alecia Willis, Patricia A Ward, Alicia Braxton, Joke Beuten, Fan Xia, Zhiyv Niu, Matthew Hardison, Richard Person, Mir Reza Bekheirnia, Magalie S Leduc, Amelia Kirby, Peter Pham, Jennifer Scull, Min Wang, Yan Ding, Sharon E Plon, James R Lupski, Arthur L Beaudet, Richard A Gibbs, and Christine M Eng. Clinical whole-exome sequencing for the diagnosis of mendelian disorders. *N. Engl. J. Med.*, 369(16):1502–1511, October 2013.

[227] Yaping Yang, Donna M Muzny, Fan Xia, Zhiyv Niu, Richard Person, Yan Ding, Patricia Ward, Alicia Braxton, Min Wang, Christian Buhay, Narayanan Veeraraghavan, Alicia Hawes, Theodore Chiang, Magalie Leduc, Joke Beuten, Jing Zhang, Weimin He, Jennifer Scull, Alecia Willis, Megan Landsverk, William J Craigen, Mir Reza Bekheirnia, Asbjorg Stray-Pedersen, Pengfei Liu, Shu Wen, Wendy Alcaraz, Hong Cui, Magdalena Walkiewicz, Jeffrey Reid, Matthew Bainbridge, Ankita Patel, Eric Boerwinkle, Arthur L Beaudet, James R Lupski, Sharon E Plon, Richard A Gibbs, and Christine M Eng. Molecular findings among patients referred for clinical whole-exome sequencing. *JAMA*, 312(18):1870–1879, November 2014.

[228] Samantha Zarate, Andrew Carroll, Medhat Mahmoud, Olga Krasheninina, Goo Jun, William J Salerno, Michael C Schatz, Eric Boerwinkle, Richard A Gibbs,

and Fritz J Sedlazeck. Parliament2: Accurate structural variant calling at scale. *Gigascience*, 9(12), December 2020.

[229] Haowen Zhang, Chirag Jain, and Srinivas Aluru. A comprehensive evaluation of long read error correction methods. *BMC Genomics*, 21(Suppl 6):889, December 2020.

[230] Aleksey V Zimin, Daniela Puiu, Ming-Cheng Luo, Tingting Zhu, Sergey Koren, Guillaume Marçais, James A Yorke, Jan Dvořák, and Steven L Salzberg. Hybrid assembly of the large and highly repetitive genome of aegilops tauschii, a progenitor of bread wheat, with the MaSuRCA mega-reads algorithm. *Genome Res.*, 27(5):787–792, May 2017.

[231] Aleksey V Zimin and Steven L Salzberg. The genome polishing tool POLCA makes fast and accurate corrections in genome assemblies. *PLoS Comput. Biol.*, 16(6):e1007981, June 2020.

[232] Justin M Zook, David Catoe, Jennifer McDaniel, Lindsay Vang, Noah Spies, Arend Sidow, Ziming Weng, Yuling Liu, Christopher E Mason, Noah Alexander, et al. Extensive sequencing of seven human genomes to characterize benchmark reference materials. *Scientific data*, 3:160025, 2016.

[233] 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.

[234] 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.

[235] 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, Sayed Mohammad E. Sahraeian, Vincent Huang, Alexandre Rouette, Noah Alexander, Christopher E. Mason, Iman Hajirasouliha, Camir Ricketts, Joyce Lee, Rick Tearle, Ian T. Fiddes, Alvaro Martinez Barrio, Jeremiah Wala, Andrew Carroll, Noushin Ghaffari, Oscar L. Rodriguez, Ali Bashir, Shaun Jackman, John J Farrell, Aaron M Wenger, Can Alkan, Arda Soylev, Michael C. Schatz, Shilpa Garg, George Church, Tobias Marschall, Ken Chen, Xian Fan, Adam C. English, Jeffrey A. Rosenfeld, Weichen Zhou, Ryan E. Mills, Jay M. Sage, Jennifer R. Davis, Michael D. Kaiser, John S. Oliver, Anthony P. Catalano, Mark JP Chaisson,

Noah Spies, Fritz J. Sedlazeck, and Marc Salit. A robust benchmark for germline structural variant detection. *bioRxiv*, 2019.

[236] 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.