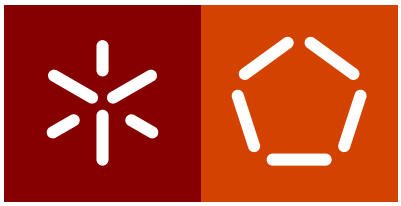
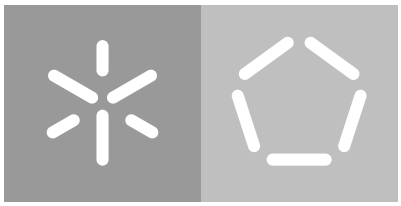


Universidade do Minho
Escola de Engenharia
Departamento de Informática

Marta Carolina Cabral Moreno

Implementation and comparison of variant calling in exome sequencing data with clinical applications

October 2018



Universidade do Minho
Escola de Engenharia
Departamento de Informática

Marta Carolina Cabral Moreno

Implementation and comparison of variant calling in exome sequencing data with clinical applications

Master dissertation
Master Degree in Bioinformatics

Dissertation supervised by
Pedro Grabriel Dias Ferreira
Miguel Francisco de Almeida Pereira da Rocha

October 2018

DECLARAÇAO

Nome
Marta Carolina Cabral Moreno

Endereço electrónico: mccm20@mailbox.org Telefone: 930661405 / —

Número do Bilhete de Identidade: 14840180

Título dissertação ☒/tese ☐
Implementation and comparison of variant calling in exome sequencing data with clinical applications

Orientador(es):
Pedro Gabriel Dias Ferreira; Miguel Francisco de Almeida Pereira da Rocha Ano de conclusão: 2018

Designação do Mestrado ou do Ramo de Conhecimento do Doutoramento:
Ramo em Tecnologias da Informação

Nos exemplares das teses de doutoramento ou de mestrado ou de outros trabalhos entregues para prestação de provas públicas nas universidades ou outros estabelecimentos de ensino, e dos quais é obrigatoriamente enviado um exemplar para depósito legal na Biblioteca Nacional e, pelo menos outro para a biblioteca da universidade respectiva, deve constar uma das seguintes declarações:

1. É AUTORIZADA A REPRODUÇÃO INTEGRAL DESTA TESE/TRABALHO APENAS PARA EFEITOS DE INVESTIGAÇÃO, MEDIANTE DECLARAÇÃO ESCRITA DO INTERESSADO, QUE A TAL SE COMPROMETE;

2. É AUTORIZADA A REPRODUÇÃO PARCIAL DESTA TESE/TRABALHO (indicar, caso tal seja necessário, nº máximo de páginas, ilustrações, gráficos, etc.), APENAS PARA EFEITOS DE INVESTIGAÇÃO, , MEDIANTE DECLARAÇÃO ESCRITA DO INTERESSADO, QUE A TAL SE COMPROMETE;

3. DE ACORDO COM A LEGISLAÇÃO EM VIGOR, NÃO É PERMITIDA A REPRODUÇÃO DE QUALQUER PARTE DESTA TESE/TRABALHO

Universidade do Minho, 26 / 10 / 2018

Assinatura: Marta Carolina Cabral Moreno

ACKNOWLEDGEMENTS

I would first like to thank my supervisor, Dr. Pedro Ferreira, for all his patience and support. He was kind enough to grant me the freedom to explore the subject matter in a way that would fit with my vision while always being there to steer me in the right direction whenever I ran into trouble. Moreover, I would like to acknowledge my co-supervisor, prof. Miguel Rocha, for providing feedback on how to improve this work.

I would also like to thank Dr. Carla Oliveira for accepting me into the Expression and Regulation in Cancer group at i3S/Ipatimup with open arms. Working in this group environment was a thoroughly enriching experienced that undoubtedly lead to my growth as a researcher and as a person. Further, I would like to thank all members of the group for their camaraderie, and for always being willing to lend a helping hand when I needed it.

Last but not least, I would like to express my utmost gratitude towards my parents. Without their boundless support and encouragement, I would have never been able to finish writing this dissertation. A huge thank you for everything.

ABSTRACT

Variant calling pipelines have been developed to help identify where and how the nucleotide bases of a genome sequence differ from its respective reference sequence. Broadly, variant calling pipelines comprise short read aligners, which align reads against a reference genome, and variant callers, which search for variants on the aligned sequences. Different read aligner and variant calling combinations have varying degrees of capability for capturing variants (recall) while reducing the amount of noise they produce (precision). Therefore, in recent years there has been an effort in comparing the performance of variant calling pipelines, although findings are far from concordant. Furthermore, some studies have postulated that the choice of sequencing technology may play a role even when all other conditions—sample, short read aligner, variant caller—remain the same.

This study aims to benchmark the performance of several variant calling pipelines on exome data sets obtained from two sequencing technologies, Illumina and Ion Torrent. To that end, variants detected in sequences obtained from a well-characterized sample, NA12878, were compared against a set of high-confidence variant calls developed for this sample using recently proposed benchmarking best practices as a basis.

This standardized framework for variant calling benchmarking replaces direct variant comparisons and Venn diagrams with more sophisticated methods. We report several informative and well-defined performance measures (precision, recall, F1 score) and present Precision-Recall curves, which are helpful for assessing pipeline performance in a visual way. Following best practices we evaluate results at the genotype match level, reporting matches only when variants are observed in the same number of alleles. The combined performance of 13 pipelines comprising three short read aligners (Bowtie 2, BWA-MEM, and TMap) and four variant callers (BCFtools, Freebayes, HaplotypeCaller, VarScan 2, and Torrent Variant Caller), was assessed on four sequencing data sets.

Our results suggest that globally variant detection appears to be independent of choice of read aligner. Overall, SNP detection performance is good for both technologies, its F1 scores averaging between 87.4% for Illumina and 84.4% for Ion Torrent. BCFtools pipelines offer the best or runner-up results for the two technologies. VarScan 2 also performs similarly well on Illumina data sets. In contrast, for indel detection, performance is very poor for Ion Torrent, with an average of 5.6% F1 score as compared to 63.1% F1 score on Illumina. The low precision levels reflected on the F1 scores reveal that there is still a long way towards improvement of indel detection, a type of variant with high impact in gene inactivation.

RESUMO

Pipelines para deteção de variantes têm sido desenvolvidas para identificar diferenças entre sequência genómicas e sequências de referência. Genericamente, as *pipelines* para deteção de variantes são constituídas por mapeadores, que localizam *short reads* num genoma de referência, e detetores de variantes, que procuram encontrar variantes nas sequências previamente mapeadas. Diferentes combinações de mapeadores e detetores de variantes possuem diferente capacidade na deteção de variantes (*recall*) e redução de ruído nos resultados (*precision*). Consequentemente, nos últimos anos tem-se comparado o desempenho destas *pipelines*, apesar destes achados ainda não serem concordantes. Ademais, alguns estudos postularam que a escolha de tecnologia de sequenciação poderá influenciar os resultados mesmo quando as demais condições—amostra, mapeador, detetor de variantes—são iguais.

Este estudo tem como objetivo avaliar o desempenho de *pipelines* para deteção de variantes quando aplicadas a conjuntos de dados exómicos sequenciados por duas tecnologias, Illumina e Ion Torrent. Deste modo, as variantes detetadas em sequências obtidas a partir de uma amostra altamente caracterizada (NA12878) foram comparadas com as presentes num conjunto de variantes de elevada confiança desenvolvido especificamente para esta amostra com base numa série de recomendações recentemente propostas.

A avaliação padronizada de desempenho substitui a comparação direta de variantes e diagramas de Venn por métodos mais sofisticados. Neste trabalho descrevemos medidas estatísticas informativas (*precision*, *recall*, e F1 *score*) e apresentamos curvas *Precision-Recall* que permitem visualizar o desemepenho das *pipelines*. Seguindo as recomendações, os resultados são avaliados ao nível do emparelhamento genómico, no qual duas variantes são consideradas iguais apenas se forem observadas no mesmo número de alelos. O desempenho combinado de 13 *pipelines* constituídas por três mapeadores (Bowtie 2, BWA-MEM, e TMap) e quatro detetores de variantes (BCFtools, Freebayes, HaplotypeCaller, VarScan 2, e Torrent Variant Caller) foi então avaliado em quatro conjuntos de dados de sequenciação.

Os nossos resultados sugerem que, no geral, a deteção de variantes é independente da escolha de mapeador. Globalmente, o desempenho na deteção de SNPs é satisfatório para ambas as tecnologias, com F1 *scores* méddios de 87.4% para Illumina e 84.4% para Ion Torrent. A ferramenta BCFTools apresenta dos melhores resultados para ambas as tecnologia, e a ferramenta VarScan 2 tem bom desempenho em dados Illumina. Por outro lado, na deteção de indels, o desempenho é muito fraco para Ion Torrent, com F1 *score* médio de 5.6% em oposição a um F1 *score* médio de 63.1% para Illumina. Os baixos níveis de *precision*

refletidos nos F1 *scores* revelam que os resultados de deteção de indels, um tipo de variante com elevado impacto na inativação de genes, carecem de grandes melhorias.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

B

BAM Binary Alignment/Map.

BCF Binary Call Format.

BED Browser Extensible Data.

BQSR Base Quality Score Recalibration.

C

CLI Command Line Interface.

E

EM Expectation-Maximization.

F

FN False Negative.

FP False Positive.

FPR False Positive Rate.

G

GA4GH Global Alliance for Genomics and Health.

GATK Genome Analysis Toolkit.

GATK-HC Genome Analysis Toolkit HaplotypeCaller.

GIAB Genome in a Bottle.

M

MNP Multi-Nucleotide Polymorphism.

N

NGS Next-Generation Sequencing.

P

PCR Polymerase Chain Reaction.

PPV Positive Predictive Value.

PR Precision-Recall.

R

ROC Receiver Operating Characteristic.

RTG Real Time Genomics.

S

SAM Sequence Alignment/Map.

SDF Sequence Data File.

SNP Single Nucleotide Polymorphism.

T

TN True Negative.

TP True Positive.

TPR True Positive Rate.

TSV Tab-separated Values.

TVC Torrent Variant Caller.

V

VCF Variant Call Format.

W

WES Whole-Exome Sequencing.

WGS Whole-Genome Sequencing.

1

INTRODUCTION

1.1 CONTEXT

In 2001, roughly a decade after its inception, the Human Genome Project released the first draft of the human genome (International Human Genome Sequencing Consortium, 2001). Having pushed Sanger sequencing to its limit, this project demonstrated the potential of genome sequencing, paving the way for the development of *Next-Generation Sequencing (NGS)* technologies that could surpass the limitations imposed by their predecessors. The NGS era brought about a great increase in data output, alongside a steep decrease in the costs associated with sequencing at a population scale, allowing researchers to gain further insight into the structure and inner workings of the human genome, as well as its protein-encoding portion, the exome, at a remarkable rate (Mardis, 2008; Metzker, 2010; Goodwin et al., 2016).

The NGS era poses new challenges, however, owning to its incessant demand for increasingly sophisticated computational pipelines capable of handling inordinate amounts of raw data while simultaneously minimizing systematic errors. One of the types of analysis birthed by NGS is variant calling, which attempts to identify differences between a genome sequence and a given reference sequence.

Although many methods have been developed for the purpose of variant calling, the results they produce are lacking in concordance (O'Rawe et al., 2013), meaning that it is important to not create new tools indiscriminately. Instead, there ought to exist a focus on refinement, with researchers measuring variant calling results against truth data sets so as to benchmark the tools' performances, as well as their aptitude to suit the needs of each type of study.

Past studies have attempted to assess the performance of variant calling tools (Liu et al., 2013; Pabinger et al., 2014; Pirooznia et al., 2014). Results produced by these benchmarking methodologies could not be compared, however, because each created its own approach in lieu of relying on a standardized framework. Therefore, in an attempt to drive the development of benchmarking standardization in the context of variant calling, the *Genome in a Bottle (GIAB)* Consortium (Zook et al., 2014, 2016) has published a set of well-characterized

genome and exome sequencing reference materials. Although still far from being a ground "truth", these high-confidence variant and reference calls have nonetheless facilitated the assessment of variant caller performance (Hwang et al., 2015; Cornish and Guda, 2015). Because of GIAB's high-confidence calls, devising pipelines for variant calling benchmarking is easier than ever before, setting the stage for the establishment of standardized comparison methods.

1.2 MOTIVATION

There is still no consensus on which variant calling pipeline provides the best results, as their performance depends not only on the choice of variant caller, but also on the type of sample (species, tissue, coverage, etc.), sequencing platform, read aligner, and so forth. Additionally, no standardized framework for benchmarking variant calling results exists, leading to studies drawing conclusions which are not only different, but also difficult, if not impossible, to compare against one another.

Previous studies have investigated what influence the choice of sequencing technology has on variant calling, but to our knowledge none has investigated the differences in results between technologies. The main reason why we think that it would be worthwhile to analyze these differences is the fact that not all technologies have the same level of community support. The sheer popularity of Illumina platforms overshadows all others, namely that of the Ion Torrent line of sequencers. Despite Ion Torrent technology being the most affected by certain types of errors and having no global error rate—which forms the basis for certain bioinformatics analyses (e.g. indel detection)—it is believed that its performance could be improved through the development of platform-specific approaches able to deal with known issues (Bragg et al., 2013). Furthermore, at the "Instituto de Investigação e Inovação em Saúde" (i3S), the institute where this research is being conducted, the genomic services have adopted Ion Torrent sequencing. Better understanding the use of this technology could lead to an optimized and streamlined variant calling pipeline suited to that technology.

The goal of this study is the implementation and comparison of different pipelines combining several short read aligners and variant callers. The results will then be compared against a high-confidence call set developed by the GIAB Consortium for that sample using a recently proposed variant calling benchmarking framework (Krusche et al., 2018). To that end, pipelines featuring several short read aligners and variant callers will be used to call variants on NA12878 *Whole-Exome Sequencing (WES)* sequences obtained from different platforms. To measure matches found in the data, a series of performance metrics will be applied. Through these metrics, we hope to be able to analyze not only the performance of the different variant callers relative to each sequencing technology, but also how results

differ between those technologies. Ultimately, these results will help inform the creation of a pipeline to be used in the analysis of clinical samples.

1.3 OBJECTIVES

Taking into account the importance of benchmarking for the optimization of variant calling pipelines, together with the influence that these pipelines exert on the analysis and interpretation of clinical results, the main objective of this work is the **implementation of variant calling pipelines to assess and compare their performance in the analysis of exome samples from multiple sequencing technologies**. To that end, a set of more specific aims will be addressed:

1. Install, try, and evaluate different short read aligners and variant callers;
2. Devise pipelines combining multiple short read aligners and variant callers;
3. Benchmark pipeline performance against a set of high-confidence variant calls;
4. Determine the strengths and weaknesses of each tool in *Single Nucleotide Polymorphism (SNP)* and indel detection;
5. Develop variant calling pipelines to analyze clinical samples.

1.4 THESIS ORGANIZATION

After this introduction, the state of the art introduces key concepts associated with genetic variation, as well as the analysis of variants in both the genome and exome. What follows is a discussion of personalized medicine, which first compares the merits and limitations of *Whole-Genome Sequencing (WGS)* and WES, and then presents statistical concepts that illustrate the need to optimize variant calling pipelines for clinical use.

Subsequent sections focus on variant calling pipelines. Section 2.3 opens with a brief look into sequencing. Section 2.4 and 2.5 introduce two fundamental components of variant calling pipelines, short read aligners and variant callers, respectively, culminating into a section which combines the two to form a general variant calling pipeline. The state of the art finishes with an introduction to variant calling benchmarking, wherein the concepts of benchmark call set and standardized benchmarking framework are explored in detail.

The methodology chapter elaborates on the various constituents of the variant calling benchmarking pipelines to be used in this work. In the interest of facilitating result reproduction, all primary inputs, methods, and tools used to build and run this work's variant calling pipelines are specified. Afterwards, we introduce *Vcaller*, a command line interface

tool that wraps around preexisting bioinformatics tools to condense the amount of lines required to run each variant calling step, and further combine those steps into automated pipelines. Lastly, we describe previous studies' findings in the field of variant calling benchmarking, and how our contribution complements and expands upon the existing literature.

In the results and discussion chapter, prediction counts and performance metrics obtained from comparing each pipeline's results against the high-confidence call set are presented in table form. Additionally, performance metrics are used to plot precision-recall curves which help visualize pipeline performance. Subsequently, these results are examined in more detail to inform the conclusions drawn in the final chapter, followed by this work's limitations and prospects for future studies in the field.

2

STATE OF THE ART

2.1 GENETIC VARIATION

Variants are genetic alterations in which one or more nucleotides differ from the reference genome at a given position or region. There are three primary types of variants (Fig. 1):

- SNPs, point mutations where one nucleotide is substituted for another (Fig. 1, top);
- Insertions and deletions (indels) occur when one or more nucleotides are added or excised between two adjacent nucleotides (Fig. 1, middle and bottom, respectively);
- Structural variants.

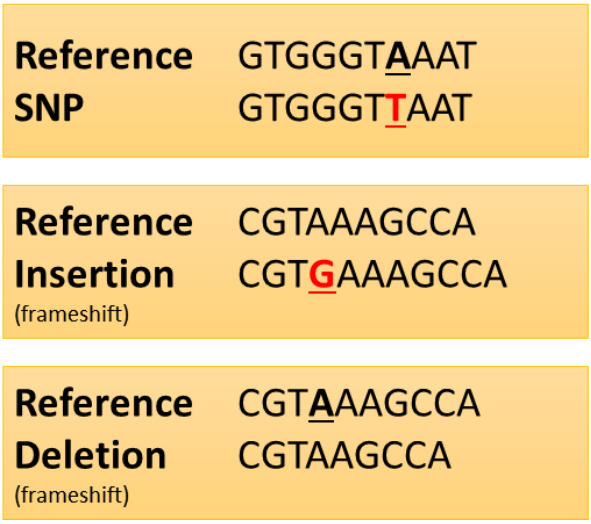


Figure 1: Example of simple genetic variants affecting a single nucleotide position.

Structural variants affect a large number of bases, meaning that, although they are abundant and biologically relevant, it is much more difficult to correctly identify them in WES data (Tattini et al., 2015). More importantly, structural variants are notably complex, and their identification requires a set of specific strategies unlike those employed for the other

types of variants. For these reasons, this work will focus on simple genetic variants: SNPs and short indels.

All individuals have variation in their genome that makes them unique. In the past, it was thought that humans shared, on average, roughly 99.9% of their genome sequence (Feuk et al., 2006), meaning that, out of the three billion nucleotides which comprise the human genome, at least three million nucleotides—one out of every 1000—would differ between individuals. Recent studies (Redon et al., 2006) have found that human variation is higher than expected—on average, in a typical human genome, 4.1 to 5 million sites differ from the reference. Of those sites, over 99.9% are SNP and indels. It is worth noting that, although fewer in number, structural variants are much more far-reaching, affecting upwards to 20 million nucleotides.

The exome may be defined as the protein-coding fraction of the genome, corresponding to roughly 1 to 2% of its total size (Warr et al., 2015). Theoretically, then, one would expect to find as few as 41 thousand and as many as 100 thousand variants in the typical human exome; however, a study which sought to determine the number of coding variants in two populations identified, on average, 20 thousand variants in European American populations, and 24 thousand variants in African American populations (Bamshad et al., 2011). This difference might be explained by the fact that regions outside the exome are less well-conserved, and thus more likely to harbor variants—or, perhaps, the continued development of bioinformatic tools will bring the number of discovered variants closer to the estimated ceiling.

2.2 THE RISE OF PERSONALIZED MEDICINE

Unlike WES, WGS can sequence entire genomes without having to limit itself to specific regions. Nevertheless, while WGS is a comprehensive approach, capable of capturing variants across exons, introns, and intergenic regions, at present little is known about the last two, meaning that exons are still more useful from a functional standpoint. More practical (albeit mostly monetary) concerns make WES an attractive option as well (Warr et al., 2015). Despite drops in the costs associated with WGS, WES remains much more affordable, making it possible to sequence exomes with higher read depth (number of reads that align to the reference genome) at a fraction of the cost. Furthermore, it may be more desirable to sequence a larger number of samples, so as to provide studies with higher statistical power. Last but not least, handling WGS data demands great amounts of computational power and storage space, which may not always be available.

These considerations become even more relevant in the clinical setting, as personalized medicine becomes commonplace (Ginsburg and Willard, 2009). For instance, exome sequencing has already been used successfully in patient diagnosis (Worthey et al., 2011).

Because of the amount of tools available—as well as the multitudes of commands, configurations, and parameters found within each one—pipelines must be optimized for a given application if they are to generate the best possible results. Therefore, it is important to build pipelines with an end goal in mind, be it clinical applications or otherwise.

To better understand the infrastructure supporting pipeline optimization, it is imperative to first explore some key statistical concepts (Parikh et al., 2008). In variant calling, when directly comparing putative variants in a query call set to a set of known variants in a benchmark call set, there are, as with any binary classification problem, four possible outcomes, often represented as a confusion matrix (Table 1):

- ***True Positive (TP)*:** A variant is called in the query call set and is also present in the benchmark call set.
- ***False Positive (FP)*:** A variant is called in the query call set but is absent from the benchmark call set.
- ***False Negative (FN)*:** No variant is called in the query call set but one is present in the benchmark call set.
- ***True Negative (TN)*:** No variant is called in the query call set and is also absent from the benchmark call set.

Table 1: Confusion matrix arising from the comparison of query calls against a benchmark set.

	Benchmark Variant	**Benchmark Reference**
Query Variant	TP	FP
Query Reference	FN	TN

TP True Positive **FP** False Positive **FN** False Negative **TN** True Negative

Performance metrics which are helpful for describing call set data can be derived from these prediction counts (Parikh et al., 2008).

- **Precision**, also known as *Positive Predictive Value (PPV)*. Measures the proportion of positive variants that are true positives. In variant calling, this metric represents the ratio of called variants also present in the benchmark call set.

$$\frac{TruePositives}{TruePositives + FalsePositives} \tag{1}$$

- **Recall**, also known as sensitivity. Measures the proportion of true positive variants from among all variants present in the benchmark call set. In other words, it measures

a call set's ability to include as many putative variants as possible, so as to avoid missing any.

$$\frac{TruePositives}{TruePositives + FalseNegatives} \tag{2}$$

For example, a general population study aiming to characterize human variation will want high recall because it seeks to capture as many variants as possible. In the clinic, the foremost priority is limiting the number of missed variants (high recall), while ensuring that the number of FPs is sufficiently low to avoid misdiagnosing patients (high precision). FNs are particularly dangerous in this context: while FPs can be validated through additional testing (e.g. Sanger sequencing), a missed variant will completely fly under the radar, which might be disastrous if the variant happened to shed light on the patient's phenotype.

2.3 THE SEARCH FOR VARIANTS

The first step in the search for variants is the sequencing of the genome or exome under study (Heather and Chain, 2016; Reinert et al., 2015). There are several platforms available, each bringing to the table its own strategy for solving the sequencing problem. Sequencing can be cycled, when molecules are incorporated one at a time, as is the case with the technologies explored in this work, or carried out in real time.

In both molecule incorporation processes, DNA is first sheared into smaller, double-stranded fragments, which can then undergo *Polymerase Chain Reaction (PCR)* amplification. Following adapter ligation, the fragments are sequenced in parallel, with strings of nucleotides being read from one or both ends of the sequence to produce single- or paired-ended reads, respectively.

In the context of Illumina sequencing, PCR amplification is mandatory, and happens within small clusters where thousands of identical molecules are being sequenced simultaneously. These molecules produce a distinct colored signal during each cycle, identifying the integrated nucleotides. Those signals are converted into arrays of nucleotides, known as reads, and stored *in silico*. For its part, Ion Torrent sequencing detects changes in pH caused by negatively or positively charged hydrogen ions released when nucleotides are incorporated into amplified fragments, meaning that changes in pH are observed only when a nucleotide matches a piece of the sheared sequence. Because the sequencer sequentially floods the chip one nucleotide type at a time, it is thus possible to measure the both the type and the quantity of incorporated nucleotides.

To limit the aforementioned process to protein-coding regions, the exome must be captured prior to sequencing (Warr et al., 2015).

In an attempt to find meaning in the haystack of reads generated by sequencing, researchers have devised heaps of algorithms capable of processing and interpreting NGS data. These algorithms can be combined into pipelines, which convey the loaded information until it has crystallized into an answer to a given biological question. At a fundamental level, the purpose of variant calling pipelines is crystal clear: uncover deviations from a reference genome in a set of NGS reads. To accomplish this goal, variant calling pipelines combine short read aligners with variant callers.

2.4 SHORT READ ALIGNERS

The genomic location of freshly sequenced reads is unknown; therefore, they must first be mapped against a reference genome. Once the bottleneck of variant calling pipelines, short read alignment, also known as read mapping, has had its efficiency pushed by developments that exploit characteristics specific to each sequencing platform, allowing it to keep up with inordinate amounts of short reads with both speed and accuracy (Li and Homer, 2010). Because incorrect alignments may confound downstream variant detection, short read aligners play a vital role in variant calling.

In broad strokes, the alignment of reads to a reference genome comprises two main steps: indexing of the reference genome (or reads), and the alignment process proper.

The indexing of the reference is usually accomplished through FM Indexing, which combines Burrows-Wheeler Transform compression (Burrows and Wheeler, 1994) with the suffix array data structure into a compressed suffix array (Ferragina and Manzini, 2000), or, alternatively, hash tables. A few examples of short read aligners comprising both algorithms for the indexing of the genome can be found in Table 2. Note that MOSAIK has not been used in this work, and is included for illustrative purposes only.

Name	Indexing Algorithm	Operating System	Citation
Bowtie 2	BWT-based FM Index	Windows, MacOSX, Linux	Langmead and Salzberg (2012)
BWA-MEM	BWT-based FM Index	Linux	Li and Durbin (2009)
MOSAIK	Hash Tables	Windows, MacOSX, FreeBSD, Linux	Lee et al. (2014)
TMap	BWT-based FM Index	Linux	N/A

Table 2: Example short read aligner tools. All tools index the genome, are able to handle paired-end reads, allow gapped alignments, and can be multi-threaded.

Notably, TMap does not implement its own mapping algorithm, but rather integrates re-implementations of various BWA algorithms optimized to run on Ion Torrent samples.

The alignment itself, which necessitates an approximate string matching paradigm, is accomplished using algorithms based on dynamic programming, namely Smith-Waterman (Smith and Waterman, 1981) for local alignments and Needleman-Wunsch (Needleman and Wunsch, 1970) for global alignments. Alignments can be gapped or ungapped; however, be-

cause re-arrangements such as indels may create gaps in the genome, gapped alignments are preferred for variant calling. Moreover, indel detection benefits from paired-end sequencing. In paired-end sequencing, fragments have adapters on the 5′ end of both strands, resulting in pairs of reads that point towards each other while leaving a gap (inner mate distance) of known size between them (Emde et al., 2012). Because the approximate length of each read is also known, paired-end sequencing can also be helpful in resolving ambiguities when one of the reads matches a repeated sequence (Reinert et al., 2015).

2.5 VARIANT CALLERS

As the name implies, variant calling plays a central role in variant detection. Therefore, a comprehensive grasp on variant calling is fundamental, and so, in addition to understanding the different types of variants (refer to Section 2.1), one should also analyze what constitutes a variant in the context of NGS (Muzzey et al., 2015).

As previously explained, reads must first be aligned to a reference genome using a short read aligner. The number of reads aligned to a specific region represents that region's read depth, and deviations from the reference genome by these reads constitute variants. Because of this concept of read depth, simple variants (SNPs and short indels) can be further divided into heterozygous and homozygous alternate.

Diploid organisms such as humans have two sets of chromosomes, meaning that each gene consists of two alleles located at the same position (locus). When most reads aligned to a given site fail to match the reference genome in the same way, alleles are said to share a homozygous alternate variant. On the other hand, heterozygous variants happen when a variant is present in only one of the alleles at that position, or, in other words, when half the reads for that position match the reference genome, while the other half differs from it. Lastly, if no variation is observed at a given position, it is referred to as homozygous reference. Therefore, variants can only be called with confidence when there are multiple reads aligned to their corresponding locus, that is, when the read depth is above a certain threshold.

Variant callers can be divided into two classes of methods: heuristic and probabilistic (Nielsen et al., 2011). Heuristic methods constitute a pragmatic strategy for problem solving, mostly based on fixed cutoffs. In the case of NGS variant calling, these methods look into heuristic factors such as minimum base quality, minimum mapping quality, and minimum coverage threshold. While heuristic variant calling methods generate adequate results, they possess severe disadvantages: the rigidity of their cutoffs can lead to the under-calling of heterozygous genotypes, and they provide no measure of confidence in and of themselves.

Probabilistic methods, on the other hand, establish frameworks based on likelihood functions which express the probability of observing a parameter given the data. Probabilistic

methods are the most widespread across all kinds of variant callers, and their most common implementation is Bayesian inference. Bayesian inference leverages Bayes' theorem, taking it as a prior model that is updated as new information becomes available.

In its simplest form, Bayes' theorem can compute the posterior probability of a genotype (G) at a certain locus with a given read count coverage (R) as follows:

$$P(G|R) = \frac{P(R|G)P(G)}{P(R)} \tag{3}$$

where $P(G)$ represents the prior probability of a given genotype—which can be assumed to be the same for all genotypes, or obtained from external sources such as the reference sequence, variant databases, and population samples—, $P(R)$ the likelihood of observing the read data at a particular site for that genotype, $P(R|G)$ the genotype likelihood, and $P(G|R)$ the posterior probability of genotype G.

Once the theorem has been used to infer the posterior probabilities for all genotypes at that locus, the genotype with the highest posterior probability is chosen and displayed alongside its degree of statistical confidence.

Below is a list of the variant callers to be explored in this work.

- **BCFTools**

  Part of the Samtools suite, BCFTools computes genotype likelihoods for each sample, one site at a time. Genotype likelihoods are determined based on the number of reads in the sample that support that site, the fraction of reference and variant bases in those reads, the error probability for each of those bases, and the reference's genotype for that site.

  For each site, the algorithm multiplies the per-sample genotype likelihoods by the binomial probability of the sample having the exact same genotype as the reference, i.e., the probability that the sample has variation at that locus. The sum of these frequencies across all samples gives an estimate for the reference allele frequency associated with a given site. An *Expectation-Maximization (EM)* optimization algorithm computes the max-likelihood estimate of the aforementioned per-site allele frequency, which is eventually converted into discrete reference allele counts. Additionally, EM optimization is used to estimate genotype frequencies at every site across all samples.

  In variant calling, it is expected that most sites are homozygous reference, meaning that they can be leveraged as priors in Bayesian inference. Thus, for each site, given the sequencing data, the distribution of allele frequencies, and the the number of reference alleles in the samples, it is possible to compute a posterior probability that is transformed into a variant's quality. If the variant quality is above a certain threshold, the site is called as variant. (Li, 2011).

- **HaplotypeCaller**

  The *Genome Analysis Toolkit (GATK)* includes HaplotypeCaller, a variant calling algorithm that searches for genome regions exhibiting peaks of activity where observed variation is expected to be above the threshold attributed to background noise. Then, HaplotypeCaller attempts to reconstruct the real sequence of each active region—in other words, its haplotype—using as input the reads that mapped to that region.

  Usually, there is more than one possible haplotype due to the diversity found in multi-sample data or eventual mapping errors, for instance. Therefore, each input read is aligned to each one of these candidate haplotypes, plus the reference sequence for that region. The resulting read-haplotype pairs are associated to a likelihood score, expressing the probability of observing the read if its companion haplotype is true.

  Next, candidate sites are examined one at a time, so that all alleles observed in the data for that position can be listed. For each read, HaplotypeCaller determines which of its associated haplotypes supports each allele; the supporting haplotype with the highest likelihood score is chosen to "represent" the read for that allele, being listed in a table with per-read likelihoods of alleles. Consequently, the likelihood of an allele being true is the product of the per-read likelihoods obtained in this way. Sites where there is sufficient evidence for at least one of the variant alleles considered will be called variant.

  Lastly, what remains is to determine the most likely genotype at each variant site. The likelihood scores of each read for each allele are leveraged as priors in Bayes' theorem to compute the probability of each possible genotype at that site; the largest probability corresponds to the most likely genotype for that site, which is thus picked (Poplin et al., 2018).

- **Freebayes**

  Unlike the two previous tools, Freebayes uses the reference genome only to obtain haplotypes, opting to call variants on literal sequences of aligned reads instead.

  For each position, Freebayes first determines potentially polymorphic regions to limit search space. Variants that are close together within these regions are grouped into haplotype allele observations. Haplotype alleles that do not meet minimum variant observations and base quality thresholds are filtered out.

  Haplotypes help determine window lengths that define the regions to genotype. Each window length is determined through an iterative process, which initially defines the window as the longest haplotype allele at that position. Observations that fully overlap this initial window are ignored; instead, the rightmost end of the longest remaining haplotype allele is used as the new starting point for the window, expanding it.

This process is repeated until none of the alleles that passed the filtering step partially overlap the current window.

After this iterative process is completed, the algorithm considers only reads found within the final window. These fully-overlapping reads are used to determine new haplotype allele observations within the window. In the end, the *a priori* genotype likelihoods of the final haplotype allele observations are passed to an expanded Bayesian model, which iterates until it converges towards the maximum *a posteriori* genotype estimate for each locus (Garrison and Marth, 2012).

- **VarScan 2**

  VarScan 2 combines heuristics with statistical testing, and takes as input *mpileup* format files, generated with Samtools (Li et al., 2009a). The *mpileup* format represents the stacking information of the reads for each nucleotide. Read base information encodes, in turn, matching status for that read if there is a SNP, insertion and deletion representation for indels, the strand the read is on, the read's mapping quality, and whether the base was found at the start of the read, end of the read, or somewhere else.

  VarScan 2 heuristically parses these files one position a time to determine how many bases from reads meeting minimum mapping quality and minimum base quality thresholds support each observed allele. The total number of such qualifying bases constitutes the coverage for that allele. From among those alleles, only those that meet a minimum coverage threshold will be considered for further analysis.

  Alleles that passed the aforementioned heuristics will then be tested in regards to whether or not they meet a minimum variant allele frequency threshold, and possess sufficient statistical significance—i.e., if their Fisher's Exact test's p-value is below a given threshold. The most frequent from among the qualifying variant alleles is reported for that position; else, if there is no such allele, the reference base is reported instead. It is important to note that variants are reported as heterozygous unless their variant allele frequency happens to be above a certain restrictive threshold (Koboldt et al., 2012).

- **Torrent Variant Caller**

  Torrent Variant Caller (TVC) is a variant caller optimized for use with data generated by Ion Torrent sequencing machines. In short, TVC attempts to find all positions with evidence for a variant, wherein it evaluates evidence for the presence of putative SNPs or indels, and then filters those candidates based on heuristic cutoffs (for base quality, coverage, and so forth), calculated through flow space evaluation.

Putative variants result from a combination of output from two sources: variants discovered using Freebayes, and long indel candidates generated by a long indel assembly module. Flow space evaluation consists in calculating the Phred-scaled posterior probability that a variant's allele frequency is above the minimum allele frequency for that variant type, which varies according to observed changes in the flow value.

In short, Varscan 2 is an example of a heuristic approach combined with statistical testing, while BCFTools, *Genome Analysis Toolkit HaplotypeCaller (GATK-HC)*, and Freebayes (and by extension TVC) are based on Bayesian statistical frameworks (Table 3).

Table 3: List of variant calling tools used in this work.

Name	Algorithmic Approach	Operating System	Citation
BCFtools	Bayesian Statistical Framework	Linux, MacOSX	Li (2011)
GATK-HC	Bayesian Statistical Framework	Linux, MacOSX	McKenna et al. (2010); DePristo et al. (2011)
Freebayes	Bayesian Statistical Framework	Linux	Garrison and Marth (2012)
Varscan 2	Heuristics combined with Fisher's Exact Test	Linux, MacOSX, Windows	Koboldt et al. (2012)
TVC	Bayesian Statistical Framework	Linux	N/A

Because this work benchmarks the performance of SNP and short indel detection in normal tissue, it considers only variant callers capable of detecting germline variants.

2.6 VARIANT CALLING PIPELINES

As explained in previous sections, short read aligners map reads to a reference genome, while variant callers compare those aligned reads to that same reference genome at every position, calling and characterizing regions where differences are detected. There are, however, complications to this simple principle: amplification biases, machine sequencing errors, software errors, mapping artifacts, and so forth. Thus, to deal with the complexity behind variant calling, the typical pipeline also encompasses quality control of the raw reads to be aligned, post-alignment processing, and, following variant calling, filtering of putative variants to be subsequently annotated (Figure 2). After exploring fundamental aspects of variant calling benchmarking, the foundations laid by this generic variant calling pipeline will be built on in the Methodology chapter.

2.7 THE BASIS FOR VARIANT CALLING BENCHMARKING

2.7.1 *Benchmark call set*

In an attempt to combat the discordance found in the calls generated by different variant calling pipelines, the GIAB Consortium is continually developing sets of high-confidence SNP, indel and reference calls obtained *in silico* through the integration of multiple methods

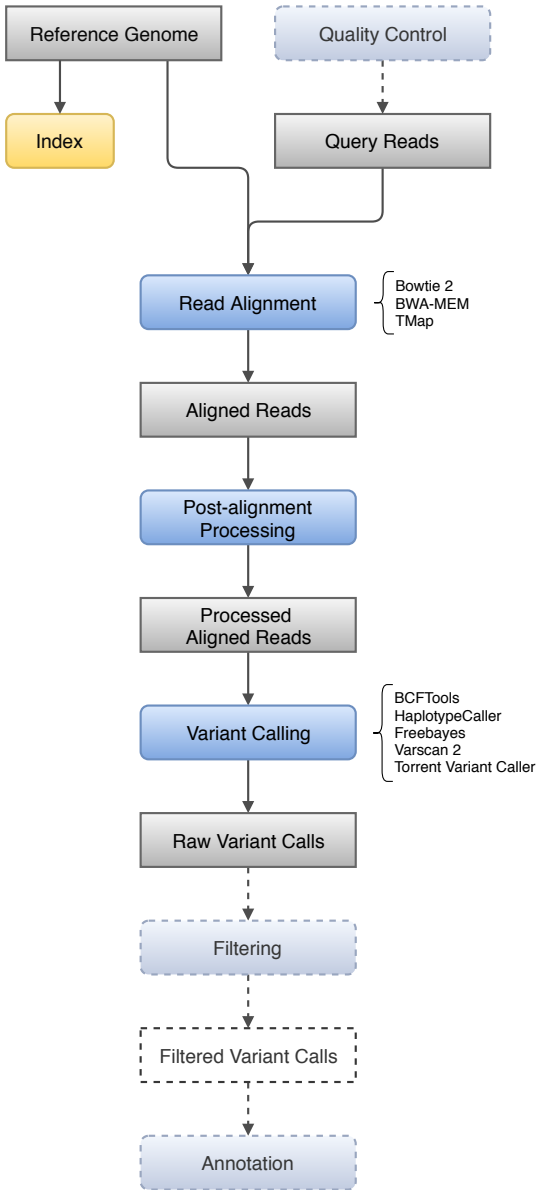


Figure 2: Example of a typical variant calling pipeline. Grayscale rectangles represent data, whereas colored rectangles with rounded corners stand for computational processes. While indispensable in a complete variant calling analysis, the quality control, filtering, and annotation steps are beyond the scope of this work, and therefore will not be explored.

and sequencing technologies (Zook et al., 2014, 2016). Although commonly referred to as "gold" standard or "truth" call set, because they are not without their uncertainties, errors, and limitations, these data sets will be interchangeably referred to as *benchmark* or *high-confidence* call sets for the remainder of this work. It is also important to note there are other high-confidence call sets available, such as Illumina's Platinum Genomes (Eberle et al., 2017).

GIAB's benchmark call sets were built on a per-sample basis using an *arbitration* method. For each sequencing technology, sensitive variant calls were obtained after running a series of variant calling tools with high-sensitivity settings, and callable regions determined following the selection of sites supported by a minimum number of reads above a given quality threshold, with subsequent exclusion of challenging regions. From among all preliminary sensitive variant calls, those supported by at least two technologies while not being contradicted by any of the other technologies were used to train a machine learning model whose goal was to find "outliers", i.e., uncertain variants associated with bias for each call set. To determine high-confidence variant calls, a new search for variants was conducted within the callable regions, with disagreements being arbitrated by the aforementioned outliers. Finally, positions containing outliers were subtracted from the union of all callable regions to form a set of high-confidence regions.

The first and most well-studied of these benchmark call sets is the one pertaining to sample NA12878, which was originally extracted from a caucasian woman from Utah to be sequenced and characterized in the context of the first phase of the 1000 Genomes Project (Auton et al., 2015). Following its adoption by GIAB, NA12878 was validated against pedigree information from her 11 offspring and their father in order to assess its accuracy.

Due to its comprehensive characterization and abundance of related materials, NA12878 was selected as the sample to be studied in this work, its high-confidence calls serving as a benchmark against which to measure the precision and recall of all calls detected in the query sequences.

Considering the workings of the call set integration, there are a series of strengths and weaknesses associated with high-confidence calls:

Strengths of high-confidence calls

- The methods used to build the benchmark call set are fully reproducible, as well as sufficiently robust to produce high-confidence calls for multiple genomes;
- Comparison against these calls is cheaper and more comprehensive than Sanger validation;
- Error-prone regions around putative structural variants are excluded from analysis;
- The usage of pedigree-based phasing provides additional evidence of whether the calls were inherited as expected in the pedigree or not.

Weaknesses of high-confidence calls

- Challenging regions are excluded, leading to underrepresentation of certain regions, such as those difficult to map with short reads, long homopolymers, tandem repeats, larger indels, and structural variants;

- Consequently, calls are biased towards both easier variants and easier genome contexts, so that precision and recall estimates may appear better than they should in some cases, and worse in others;
- For each technology, reads are passed through a single "optimal" variant calling pipeline.
- In spite of its narrower scope, Sanger sequencing still constitutes the most reliable form of variant validation.

2.7.2 *A standardized framework for benchmarking*

The existence of benchmark call sets raises the question of how to best leverage their high-confidence calls for the purposes of variant calling benchmarking. For a few years, there was no consensual answer to this question, and so most studies would take on their own novel approaches. In recent times, however, the *Global Alliance for Genomics and Health (GA4GH)* has created a Benchmarking Team whose goal is to drive the standardization of variant calling benchmarking. As a result of this team's efforts, the GA4GH has proposed a set of best practices, or standardized framework, for utilizing existing reference materials and tools for benchmarking, as well as details on how to best interpret benchmarking results.

The gist of their standardized framework goes as follows: for each comparison, a benchmark call set and a query call set are passed as input to a comparison engine, which will produce an intermediate call set containing matches to be quantified. With this standardized framework, the GA4GH aims to overcome the following challenges:

1. Variants cannot be compared directly due to representation differences in benchmark versus query data sets.
2. Comparison metrics must be standardized and well-defined if they are to support robust conclusions.
3. Performance is dependent on variant type (e.g. SNPs vs indels) and genome context, creating a need for stratification.

Variant Representation

Different variant calling methods may produce different variant representations for calls in the *Variant Call Format (VCF)*. For example, variant callers may opt to either group a *Multi-Nucleotide Polymorphism (MNP)* into a single call or break it into individual SNPs, which would be perceived as distinct variants when performing direct comparison at each

position. Therefore, benchmarking VCF files by comparing alternate allele records directly can lead to erroneous results (Table 4).

	Chr	Pos	Ref	Alt
BCFtools	1	1301807	**A**~~GTGTGATTGAATGAGT~~	**A**
HaplotypeCaller	1	130807	**A**~~GTGTGATTGAATGAGT~~**GTG**	**AGTG**

Table 4: Example of how representation differences between two variant callers for reads from the same sequence aligned by the same read aligner can lead to "different" variants in the case of a homopolymer deletion.

Chr Chromosome **Pos** Position **Ref** Reference Allele **Alt** Alternate Allele

To address the issue of variant representation, it is imperative that the comparison between benchmark and query call sets is independent of how each variant appears in the VCF, which requires normalization. There are various methods and tools for dealing with this problem; in this work, the chosen solution was Real Time Genomics' comparison tool *vcfeval* (Cleary et al., 2015).

Briefly, to normalize representation, *vcfeval* "replays" variants from the benchmark and query call sets back to the reference genome. At any given locus there might exist more than one possible normalized representation, however, and for this reason *vcfeval* leverages global optimization to select the most parsimonious of those representations.

First, *vcfeval* replays variants against the reference genome, one at a a time. During this replay, nucleotides are taken from either the reference genome or the variant, and used to build sets of subsequences associated with either the benchmark call set or the query call set, meaning that each variant may be represented by more than one subsequence. The pair of subsequences which maximizes the similarity between the benchmark and query call sets is chosen to represent that variant.

Armed with this set of variant subsequence pairs, *vcfeval* can proceed to create benchmark and query paths, each comprising a distinct subset of variants. After the paths have been built, benchmark and query path pairs are selected with the goal of maximizing the number of variants they share (TPs). In order to increase path similarity, calls may have to be excluded from the benchmark path (FNs) or from the query path (FPs). Multiple path pairs are created for each variant subset, but only the pair which maximizes TPs and minimizes FNs and FPs is chosen in the end.

Like all methods described in this section, *vcfeval* comes with its share of strengths and weaknessess:

Strengths of *vcfeval*

- Solves the issue of inconsistent variant representation.

- More informative than the metrics and Venn Diagrams obtained from direct comparison.
- Analysis of *Precision-Recall (PR)* curves allows the user to reach a desired precision/recall balance.

Weaknessess of *vcfeval*

- Does not follow the standardized comparison metric definitions proposed by the GA4GH.
- Result metrics cannot be stratified.

Nevertheless, it is possible to compensate for *vcfeval*'s weaknessess by leveraging one of its output modes, `-m ga4gh`, which produces a GA4GH-intermediate VCF file. As will be discussed below, this file serves as input for a quantification tool developed by the GA4GH Benchmarking Team, *qfy.py*, which is a part of the Haplotype Comparison Tools suite (`https://github.com/Illumina/hap.py`).

Performance Metrics and Quantification

Because of the inherent complexity of the human genome, TP, FP, and FN matches can be defined in different ways, depending on matching stringency between benchmark and query call set—in other words, the high-confidence call set from GIAB and the call set whose performance is to be measured.

- **Local Match**: A variant discovered in the query call set is within a given local matching distance to where another variant was present in the benchmark call set.
- **Allele Match:** Both query and benchmark call sets have detected the same variant allele; in other words, their ALT nucleotides match after variant representation is normalized.
- **Genotype Match:** The query call set and the benchmark call set not only share a variant allele, but also a variant genotype, meaning that the variant is present in the same number of alleles in both sets, although the variant's location on those alleles (phasing) is unknown.
- **Phased Genotype Match:** Similar to the genotype match, but now information concerning the phasing is considered; therefore, the exact alleles where each variant is located at are known. Phased genotype matches only occur when both benchmark and query variant alleles are in the same strand(s).

Table 5: Matching example for the allele, genotype, and phased genotype match types. Variation (ALT) relative to the reference genome (REF) has different possible genotypes for each of two data sets (GT Benchmark and GT Query), which can be phased (|) or unphased (/). A graphical representation of these possibilities can be found in Figure 3.

	REF	ALT	GT Benchmark	GT Query
Allele Match	G	T	0/1	0/1, 1/0 or 1/1
Genotype Match	G	T	0/1	0/1 or 1/0
Phased Match	G	T	0|1	0|1

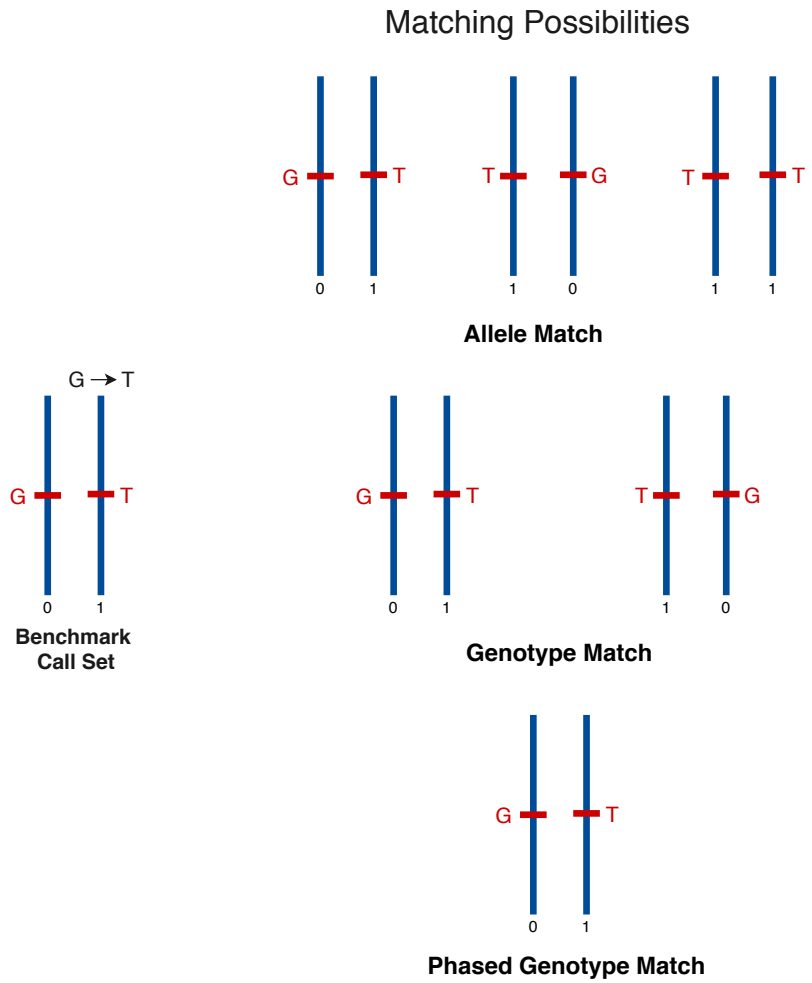


Figure 3: Example of matching possibilities for the allele, genotype, and phased genotype match types, provided that the benchmark exhibits a G→T transversion and a 0/1 genotype; refer to Table 5 for details.

An example of how matching works in the three more stringent cases should the benchmark set exhibit a G→T transversion and a 0/1 genotype is shown using VCF representation in Table 5, and graphically in Figure 3.

This work will focus on genotype matching, the default stringency used by GA4GH-compliant tools. Therefore, performance metrics used to describe how putative variants in the query call set match against those in the benchmark call set will differ from the more general ones discussed in Section 2.2:

- **True Positive (TP):** A variant genotype is called in the query and that same genotype is present in the benchmark call set.
- **False Positive (FP):** A variant genotype is called in the query but that same genotype is absent from the benchmark call set.
- **False Negative (FN):** No variant genotype is detected in the query but one is present in the benchmark call set.

One might notice that a commonly used performance metric, TN, is missing from the above list. This happens because it is difficult to define TNs in the context of the genome: because a very large number of reference alleles exists in the genome, there is likewise a very large number of TNs. For this reason, going forward performance metrics dependent on TNs will be absent from this study.

Statistical measures derived from these key performance metrics also vary in accordance with matching stringency. In the context of genotype matching, they are defined as follows:

- **Precision:** Fraction of query calls present in the benchmark call set, translating into the query set's ability to limit calls to only those present in the benchmark calls. Because they are absent from the benchmark call set, FPs are determined relative to the query call set; therefore, they are compared against query TPs, which have the same variant representation.

$$\frac{Query.TP}{Query.TP + Query.FP} \tag{4}$$

- **Recall:** Fraction of benchmark calls present in the query call set, translating into the query set's ability to detect variants known to be present in the benchmark call set. Because they are absent from the query call set, FNs are determined relative to the benchmark call set; therefore, they are compared against benchmark TPs, which have the same variant representation.

$$\frac{Benchmark.TP}{Benchmark.TP + Benchmark.FN} \tag{5}$$

- **F-Score:** The harmonic mean between precision and recall.

$$\frac{2}{\frac{1}{Recall} + \frac{1}{Precision}} \Leftrightarrow 2 \times \frac{Recall \times Precision}{Recall + Precision} \tag{6}$$

For the quantification step, the aforementioned *qfy.py* script can be used to tally prediction counts for variant matches in a GA4GH-intermediate file, as well as compute performance metrics such as those described above. Like equations 4 and 5 suggest, *qfy.py* counts TPs for both query and benchmark call sets, FPs for the query call set, and FNs for the benchmark call set. Moreover, this tool is capable of transforming prediction counts and related prediction metrics into points that can be plotted as a PR curve. It also has the ability to stratify matches beyond separating SNPs from indels and other complex variants through options such as `--stratification`, which takes a stratification file list in the *Tab-separated Values (TSV)* format.

Armed with the means to solve the issues of 1) variant representation, 2) standardization of prediction counts and performance metrics, and 3) variant stratification, it is time to search for reads sequenced from the same sample as the benchmark call set, so as to pass them through variant calling pipelines, and move on to variant calling benchmarking proper.

2.8 PREVIOUS WORK

In order to assess the relevance and novelty of the approaches explored in this work, its main features were compared to those of previous studies in the field of germline variant calling benchmarking, as summarized in Table 6.

Table 6: Comparison of features studied across germline variant calling benchmarking works.

	Compares Aligners and Variant Callers	Stratifies SNPs and indels	Compares Sequencing Technologies	Normalizes Representation	High-confidence Call Set
Pabinger et al. (2014)	×	×	×	×	×
O'Rawe et al. (2013)	×	√	√	×	×
Liu et al. (2013)	×	×	×	×	×
Yu and Sun (2013)	×	×	×	×	×
Bao et al. (2014)	√	√	×	√	√
Cornish and Guda (2015)	√	√	×	√	√
Hwang et al. (2015)	×	√	×	√	√
Laurie et al. (2016)	√	√	×	√	√
Moreno et. al 2018	√	√	√	√	√

In spite of having a later publishing date, the work of Pabinger et al. (2014) was first made available online in January of 2013. Its primary focus was to survey various bioinformatics tools to help researchers build their own workflows for NGS variant detection. With that goal in mind, Pabinger et al. (2014) also evaluated the results of the suggested primary steps, including variant calling. Their comparison was limited to a simple Venn diagram showcasing variant concordance among the five tested variant callers, with no segregation between SNPs and indels.

Venn diagrams would go on to become a staple of early variant calling benchmarking studies. Later works by O'Rawe et al. (2013), Liu et al. (2013), and Yu and Sun (2013) continued the trend of only measuring variant caller performance, pairing them with their respective state-of-the-art short read aligners. While the work of Yu and Sun (2013) analyzed only SNPs and that of Liu et al. (2013) paired SNPs and indels under the "variants" denomination, O'Rawe et al. (2013) embraced stratification, comparing the two simple variant types separately. In all cases, benchmarking was limited to representing the number of shared variants in Venn diagrams. Additionally, O'Rawe et al. (2013) measured the concordance of SNP and indel calling between two technologies (Illumina and Complete Genomics), also through Venn diagrams.

SMASH (Talwalkar et al., 2014) is a variant calling benchmarking toolkit first released following these initial strides in the development of germline variant calling benchmarking, providing researchers with well-defined methodologies (such as variant normalization) and evaluation metrics to standardize comparison results. Leveraging this toolkit, Bao et al. (2014) surveyed multiple bioinformatics tools used in the context of WES variant analysis, and evaluated their performance. More importantly, this study's comparisons comprised not only variant callers, but short read aligners as well. Although it still presented Venn diagrams for call concordance, this study also used SMASH to compare its results against the benchmark call sets provided by the GIAB Consortium so as to measure variant calling performance.

More recent studies include those of Cornish and Guda (2015) and Hwang et al. (2015), which combined multiple short read aligners and variant callers to generate their calls. Note, however, that Hwang et al. (2015) did not measure read aligner performance, but rather integrated calls from different variant callers on a per read aligner basis, meaning that only the former were truly evaluated. As was the case with the work discussed above, these studies also compared their results against the GIAB high-confidence call set for sample NA12878.

Cornish and Guda (2015) used only one exome data set, while Hwang et al. (2015) used a large number of data sets from two sequencing technologies (Illumina and Ion Torrent) to avoid data-related biases in their results. Both works took the time to define the prediction counts and performance metrics applied to their results, although there was no consensus

on nomenclature. To solve the variant representation issue, these works performed variant normalization through vcflib's *vcfallelicprimitives* (`https://github.com/vcflib/vcflib`).

These two studies still included Venn diagrams as a way to measure variant calling concordance, but it became apparent that this methodology had reached a bottleneck: because drawing information from Venn diagrams becomes too complicated when the number of classes of objects (circles) is too large, Cornish and Guda (2015) could only represent five out of their 30 pipelines. Therefore, for the first time in the context of this field of study, Hwang et al. (2015) leveraged PR curves as a benchmarking metric.

Lastly, Laurie et al. (2016) continued treading the same path as the studies of Cornish and Guda (2015) and Hwang et al. (2015), but with an additional focus on comparing WGS and WES, and the differences in the computational costs of each tool.

To further progress research done in the field, we have integrated features from previous works, solutions to some of the issues and limitations they have encountered, and novel approaches proposed in the best practices developed by pioneers in the field (Krusche et al., 2018) which seek to increase the robustness and reproducibility of variant calling benchmarking (Table 6).

Our study continues the trend of comparing the performance of both short read aligners and variant callers, while distinguishing between results for SNPs and indels, given that the performance of variant detection methods for each type of variant has been proven to be significantly different. Furthermore, similarly to other recent studies, we compare calls against a high-confidence call set after normalizing their variant representation.

To our knowledge, our study is the first to compare Illumina and Ion Torrent sequencing technologies and to adopt the variant calling benchmarking best practices proposed by the GA4GH. In addition, we use PR curves instead of Venn diagrams to represent variant calling benchmarking results, because the latter provide a limited amount of information and become difficult to interpret when there is a large number of comparisons (circles).

3

METHODS

The variant calling benchmarking pipelines presented in Fig. 4 are the culmination of all the tools and techniques surveyed thus far. The post-alignment processing step shared by those pipelines is detailed in Fig. 5. For the remainder of this section, the pipelines' inputs, methods, and outputs will be discussed in length.

3.1 PRIMARY INPUTS

What follows is a list of the primary inputs passed to the variant calling benchmarking pipelines at various points, their characteristics, the reasoning behind their choice, and any preprocessing steps that needed to be performed.

3.1.1 *Reference genome*

The main reference genome used in this work was the 1000 Genomes Project's version of the GRCh37 human genome assembly, build b37. This build was chosen partially because it is included in a resource bundle available on the Broad Institute's FTP server (`ftp://gsapubftp-anonymous@ftp.broadinstitute.org/bundle/b37/`), which contains not only the FASTA file pertaining to build b37, `human_g1k_v37.fasta`, but also other useful files such as lists of known SNPs and indels. More importantly, the Broad Institute recommends build b37 for WES variant calling, and the GrCH37 GIAB high-confidence call set (compatible with b37) still covers a higher percentage of non-N bases than the corresponding GrCH38 call set as of version 3.3.2 (Zook et al., 2018).

It was also necessary to obtain the human genome build used in the alignment of the prealigned Ion Proton GIAB data set, hg19 (UCSC's version of GRCh37 reference assembly), so as to leverage its *Binary Alignment/Map (BAM)* file in TVC variant calling. Because the reference FASTA file contained within the Broad Institute's hg19 bundle was not fully compatible with Ion Proton GIAB's BAM file, likely due to patching differences, the independent chromosome sequences were instead directly obtained from the UCSC

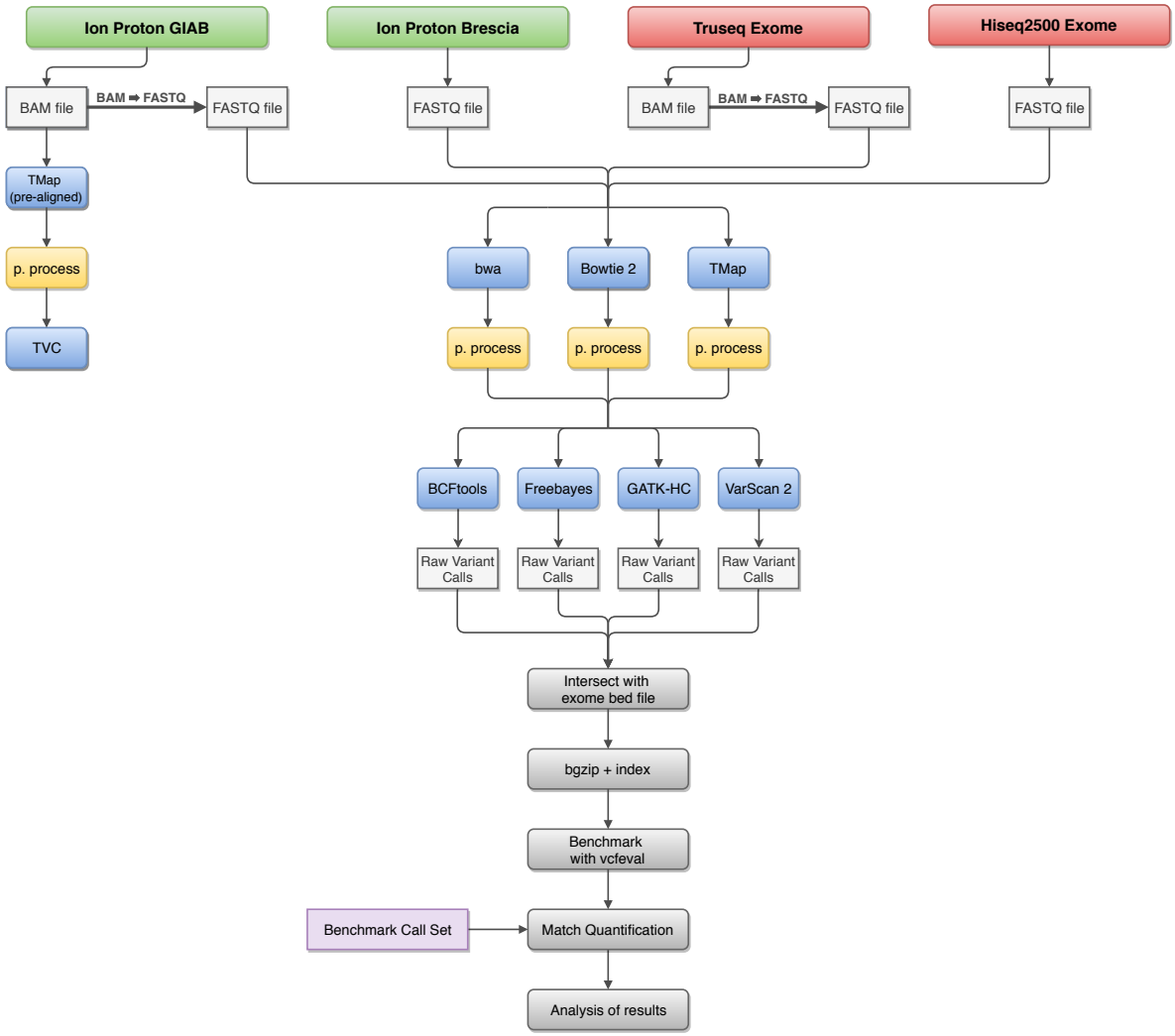


Figure 4: Pipelines for variant calling benchmarking to be used in this work. Refer to Fig. 5 for more details on the post-alignment processing (p. process) step.

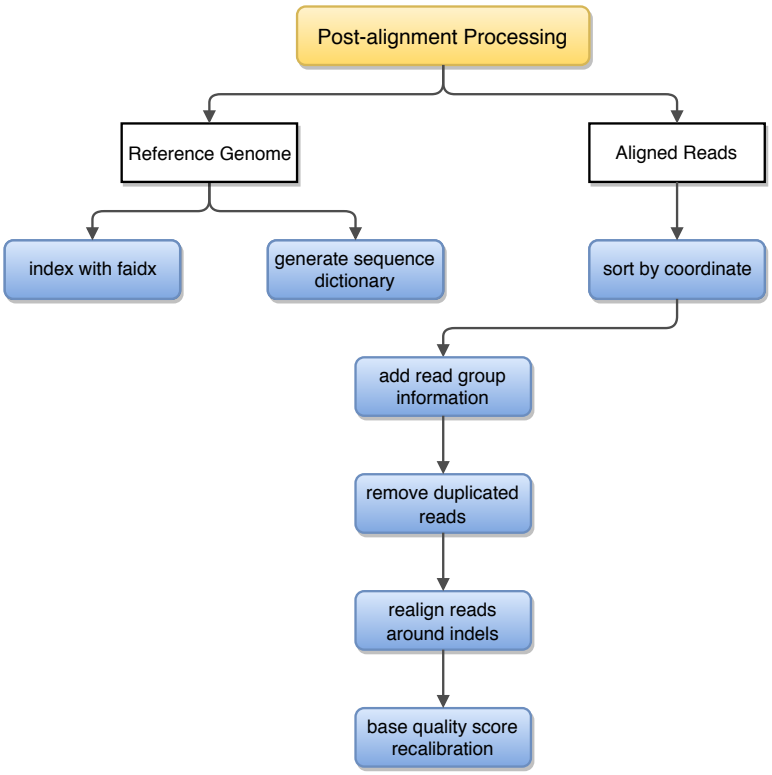


Figure 5: Details of the post-alignment processing pipeline used in Fig. 4.

(http://hgdownload.cse.ucsc.edu/goldenPath/hg19/bigZips/). In addition, the hg19 build against which this Ion Proton data set had been aligned used a revised mitochondria sequence; for this reason, the UCSC mitochondria sequence file `chrM.fa` was replaced with a newer version found under GenBank record NC_012920. Lastly, all chromosome sequences were merged into a single FASTA file, `hg19.fa`.

3.1.2 *Query reads*

As previously established, variant calling requires BAM files (data sets) containing query reads which have been aligned against a reference genome. Reads cannot be chosen carelessly, especially if one is to benchmark sequencing technology performance; therefore, certain criteria had to be met by query reads and prealigned query reads for them to be explored in this work:

- The reads must have been sequenced from sample NA12878, so that they may be compared to the NA12878 benchmark call set;
- Further, reads must have been sequenced using WES;
- Reads had to be sequenced using either Ion Torrent or Illumina technologies.

Table 7 summarizes the characteristics of the data sets chosen for this study, while table 8 lists the repositories from which these data sets were obtained.

Table 7: Characterization of the data sets used in this work.

Data Set	Technology	Sequencer	Exome Kit	# Runs
Ion Proton GIAB	Ion Torrent	Ion Proton	AmpliSeq Exome	1
Ion Proton Brescia	Ion Torrent	Ion Proton	AmpliSeq Exome	4
Illumina TruSeq	Illumina	Genome Analyzer IIx	TruSeq Exome	1
Illumina HiSeq2500	Illumina	HiSeq2500	Nextera Rapid Capture Exomes	4

Table 8: Repositories containing the data sets used in this work.

Data Set	Repository URL
Ion Proton GIAB	ftp://ftp-trace.ncbi.nlm.nih.gov/giab/ftp/data/NA12878/ion_exome/
Ion Proton Brescia	https://trace.ncbi.nlm.nih.gov/Traces/study/?acc=SRP109084
Illumina TruSeq	ftp://ftp-trace.ncbi.nlm.nih.gov/giab/ftp/data/NA12878/Nebraska_NA12878_HG001_TruSeq_Exome/
Illumina HiSeq2500	ftp://ftp-trace.ncbi.nlm.nih.gov/giab/ftp/data/NA12878/Garvan_NA12878_HG001_HiSeq_Exome/

Both Ion Proton GIAB and Illumina TruSeq were sequenced in a single run (lane), whereas Ion Proton Brescia and Illumina HiSeq2500 were sequenced in four runs. Moreover, Ion Proton Brescia and HiSeq2500 were obtained as unaligned query reads, i.e. FASTQ files. On the other hand, Ion Proton GIAB and Illumina TruSeq were made available as prealigned BAM files. These BAM files were converted into FASTQ files using the command `bedtools bamtofastq` so that they too could serve as query reads. It is important to note that Ion Proton GIAB's pre-aligned query reads were also used in this work, because they contain the flow signal information required by Ion Torrent's variant caller, TVC, which is lost in FASTQ conversion.

The *Browser Extensible Data (BED)* files corresponding to the regions captured by the AmpliSeq Exome and TruSeq Exome kits were originally built for the hg19 build of the human reference genome, meaning that they had to be edited in order to comply with the chromosomal nomenclature found in build b37. This was accomplished through the following command:

```
$ sed 's/chr//g' hg19_exome_kit.bed > b37_compliant_exome_kit.bed
```

One important caveat of this analysis is that we could only find one prealigned Ion Torrent BAM file with flow signal information freely available online. Furthermore, this data set was among those used to build the GIAB high-confidence call set, analyzed using the same short read aligner (TMap) and variant caller (TVC) combination as this work. Because of the strong potential for bias, it was imperative to search for additional Ion Torrent query reads, but the only other data set concerning sample NA12878 that we could

find, Ion Proton Brescia, contained no flow information; therefore, the TMAP + TVC pipeline was limited to Ion Proton GIAB.

3.1.3 *Benchmark call set*

What are benchmark call sets, how are they built, and what are their strengths and limitations are questions that have been answered in Section 2.7.1. The focus of this work lies on the benchmark call set pertaining to sample NA12878. Its latest version can be found in GIAB's FTP (`ftp://ftp-trace.ncbi.nlm.nih.gov/giab/ftp/release/NA12878_HG001/latest/GRCh37/`). When this study was underway, the latest version available was v3.3.2. GIAB provides not only the benchmark call set and its index, but also a BED file describing the high-confidence regions in which the query and benchmark call sets are to be compared.

3.1.4 *Lists of known variants*

The b37 bundle obtained from the Broad Institute's FTP server contained a few lists of known variants, described in Table 9, that were used in certain variant calling steps.

Table 9: Lists of known variants and their role in this work.

Know Variants File	**Used For**
`1000G_phase1.indels.b37.vcf.gz`	indel realignment and base score recalibration
`Mills_and_1000G_gold_standard.indels.b37.vcf.gz`	indel realignment and base score recalibration
`dbsnp_138.b37.vcf.gz`	base score recalibration and HaplotypeCaller

3.2 VARIANT CALLING PIPELINES: TOOLS AND METHODS

Section 2.7.2 laid out the foundations for a typical variant calling pipeline, which was built on to establish the variant calling pipelines used in this work (Fig. 4). All commands used to run the operations described in this section are detailed in supporting material A.1.

Similar to the more general variant calling pipeline in Fig. 2, the variant calling pipelines crafted for this work were based on the guidelines laid out by the Genome Analysis Toolkit's Best Practices (Van der Auwera et al., 2013). The pipelines comprise seven main steps: read alignment, post-alignment processing of the aligned reads, variant calling, post-processing of the resulting call set, call set benchmarking against a high-confidence call set, quantification of the observed matches, and analysis of the final results.

In the interest of facilitating reproducibility, Table 10 contains a list of all software supporting this study's variant calling pipelines. Note that, when possible, tools were run with default settings.

Table 10: List of software used in this work.

Tool	Version
BCFTools	1.4.1+htslib-1.4.1
bedtools	v2.25.0
Bowtie 2	2.3.4.1
BWA-MEM	0.7.15-r1140
Freebayes	1.2.0
GATK 3	3.8-0-ge9d806836
GATK 4	4.0.2.1
Samtools	1.2+htslib 1.2.1
VarScan 2	2.4.3

GATK 3: Required for a post-processing step, indel realignment.
GATK 4: Used for general post-processing and variant calling with HaplotypeCaller.

3.2.1 *Read alignment*

Read alignment requires two inputs: a reference genome, and reads to align against it. A list of all read aligners used in this work, as well as details on their inner workings, can be found in Section 2.4.

Most reads were aligned separately, including those sequenced in more than one lane, namely the single-ended runs of Ion Proton Brescia. Illumina HiSeq2500 was the only data set in this work with paired-end reads, which were aligned simultaneously.

3.2.2 *Post-alignment processing*

Post-alignment processing steps are summarized in Fig. 5.

Read group information, which consists of tab-separated metadata concerning the query reads, was added to the aligned sequences to facilitate further downstream analysis: a unique ID, the sample name, library identifier, sequencing platform, and sequencing platform unit (e.g. run barcode).

PCR amplification ensures the existence of a sufficient amount of DNA in downstream steps, and is therefore required for library preparation. Duplicate reads originating from the PCR amplification step artificially increase the number of reads supporting certain variants, which could influence their detection (Li et al., 2009b). Furthermore, these duplicate

reads needlessly increase the number of reads that need to be processed, increasing the computational cost of all downstream steps. Therefore, duplicated reads were marked and subsequently removed from all data sets.

Moreover, short read aligners might place reads on the edges of indels, creating mismatches between bases, and thus leading to the incorrect calling of SNPs. Realignment around known indels is a two-step process that aims to combat this issue. First, a list of intervals in need of intervention was generated, and that list was then passed to a command able to perform the realignment proper.

NGS platforms produce reads in the FASTQ format, which combines FASTA sequences with their associated per-base Phred-scaled base quality scores. Base quality scores express a sequencer's confidence in its calling the correct base at each position, consequently making it possible to evaluate sequencing machine errors. The emission of base quality scores can be subject to systematic errors, however, negatively impacting their usefulness. Thus, a machine learning approach capable of detecting systematic errors so as to model them and readjust base quality scores, *Base Quality Score Recalibration (BQSR)*, was run on all data sets.

3.2.3 *Variant calling*

Variant calling requires two inputs: a reference genome, and a set of aligned (and, preferably, processed) reads. A list of all variant callers used in this work, as well as details on their inner workings, can be found in Section 2.5.

3.2.4 *Variant calling benchmarking*

Table 11: Read aligner and variant caller combinations which constitute the pipelines used in this work.

	BCFTools	**Freebayes**	**HaplotypeCaller**	**VarScan 2**	**TVC**
Bowtie 2	Bowtie 2 + BCFTools	Bowtie 2 + Freebayes	Bowtie 2 + HaplotypeCaller	Bowtie 2 + VarScan 2	N/A
bwa	BWA-MEM + BCFTools	BWA-MEM + Freebayes	BWA-MEM + HaplotypeCaller	BWA-MEM + VarScan 2	N/A
TMap	TMap + BCFTools	TMap + Freebayes	TMap + HaplotypeCaller	TMap + VarScan 2	TMap + TVC

Each of the four samples went through three read aligners × four variant callers pipelines, plus the `TMap + TVC` pipeline which was run exclusively on the Ion Proton GIAB data set, for a total of 13 distinct pipelines (described in Table 11). In total, therefore, 49 query call sets were compared against the chosen benchmark call set using *vcfeval*, resulting in 49 comparisons and their respective metrics.

The Linux version of *vcfeval* was obtained from the *Real Time Genomics (RTG)* website (`https://www.realtimegenomics.com/products/rtg-tools`). For the quantification step,

qfy.py, a part of the Haplotype Comparison Tools suite (https://github.com/Illumina/hap.py), was run on the raw benchmarking metrics.

3.3 VCALLER

Variant calling, while simple in principle, can quickly become a daunting task to the average user, given that, as previously shown, the typical pipeline requires back-and-forth usage and configuration of multiple command line tools. To simplify this task in the context of variant calling benchmarking, we have developed *Vcaller* (https://github.com/martaccmoreno/vcaller), a *Command Line Interface (CLI)* capable of invoking multiple pre-existing bioinformatics tools, eclectically grouping them into commands that will perform each of the main steps described above (refer to Fig. 6). Therefore, because *Vcaller* is composed by these intuitive building-block commands, it retains enough flexibility to allow the user to build their own variant calling pipelines without having to worry about the minutia inherent to the usual "mix-and-match" approach commonly observed in the field.

To illustrate *Vcaller*'s reduction abilities, below are the commands used to run the `BWA-MEM + BCFtools` benchmarking pipeline on the Ion Proton GIAB data set through *Vcaller*.

```
1 $ vcaller align bwa -o ion_bwa.bam b37/human_g1k_v37.fasta
    IonXpress_020_rawlib.b37.fastq.gz
2 $ vcaller process -o ion_bwa_processed.bam --readgroup-info ID:020,
    PU:PGM/P1.1.17/IonXpress_020,PL:IONTORRENT,SM:NA12878,LB:lib1 --
    add- known-indels b37/1000G_phase1.indels.b37.vcf.gz b37/
    Mills_and_1000G_gold_standard.indels.b37.vcf.gz b37/dbsnp_138.
    b37.vcf.gz human_g1k_v37.fasta ion_bwa.bam
3 $ vcaller call bcftools -o ion_bwa_bcftools.vcf --exome-regions
    AmpliseqExome.20141120_effective_regions.b37.bed b37/
    human_g1k_v37.fasta ion_bwa_processed.bam
4 $ vcaller compare --output-dir ion-bwa-bcftools --bed-file data/
    exome-regions/AmpliseqExome_nochr.bed --evaluation-regions giab/
    NA12878_GIAB_highconf_CG-IllFB-IllGATKHC-Ion-Solid-10X_CHROM1-
    X_v3.3_highconf.bed b37/human_g1k_v37.fasta giab/
    NA12878_GIAB_highconf_CG-IllFB-IllGATKHC-Ion-Solid-10X_CHROM1-
    X_v3.3_highconf.vcf.gz ion_exome_bwa_bcftools.vcf
```

Listing 3.1: *Vcaller* commands used to run the BWA-MEM + BCFTools pipeline according to the steps delineated thorough this section.

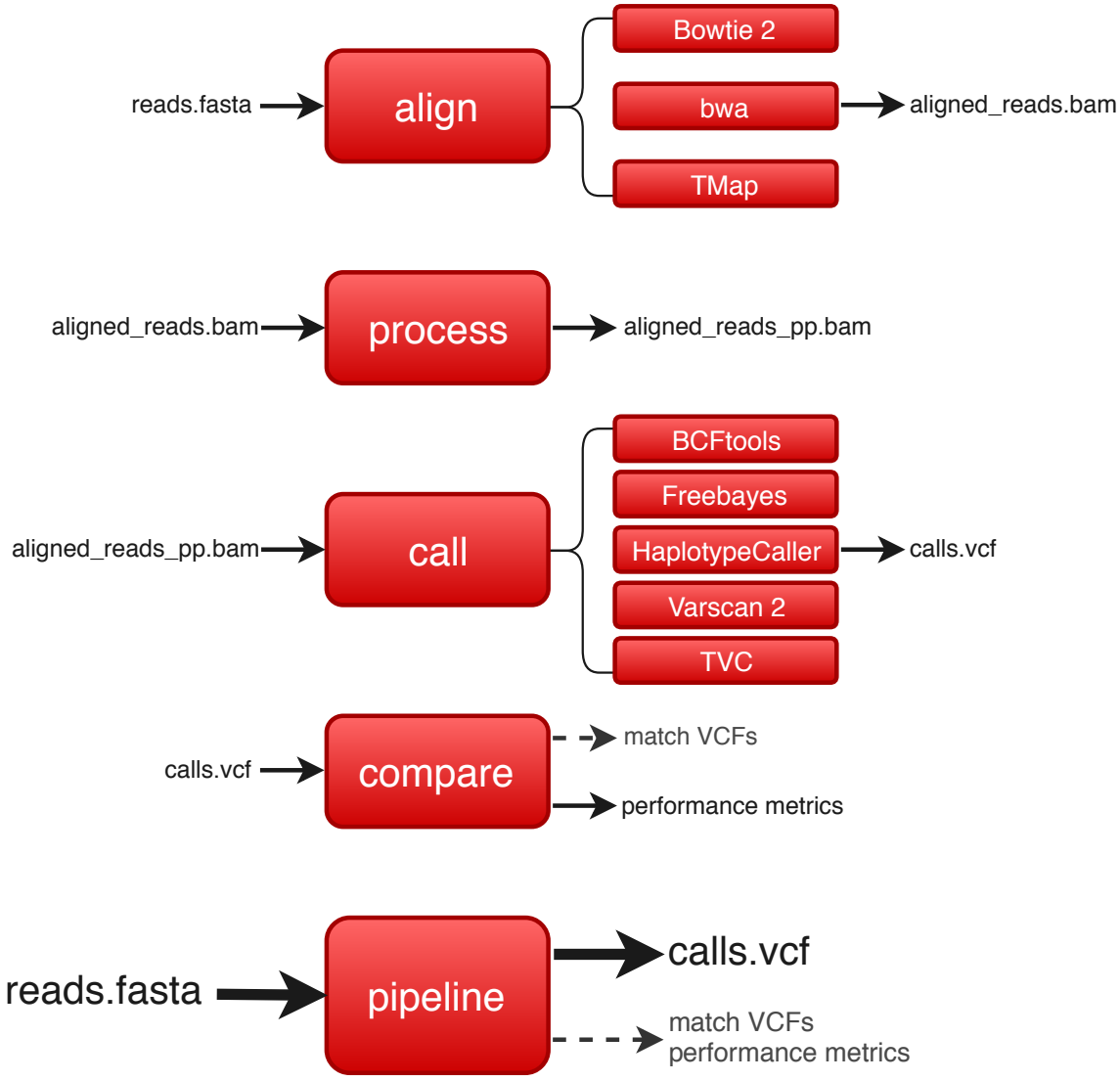


Figure 6: All main *Vcaller* commands and subcommands. The `align`, `process`, `call` (and, optionally, `compare`) commands can be sequentially run to obtain a result similar to that of the `pipeline` subcommand. When running `vcaller pipeline`, the user must define arguments to choose what read aligner and variant caller combination to use. The pipeline can either be a variant calling pipeline, or, if an option is toggled, a variant calling benchmarking pipeline similar to those described in this work. Most of the intermediary files (e.g., VCFs containing benchmarking match information) are removed by default, but can optionally be kept.

Or, more simply:

```
$ vcaller pipeline bwa bcftools b37/Mills_and_1000G_gold_standard.
    indels.b37.vcf.gz b37/1000G_phase1.indels.b37.vcf.gz b37/
    human_g1k_v37.fasta IonXpress_020_rawlib.b37.fastq.gz -o
    ion_bwa_bcftools.vcf --exome-regions AmpliseqExome.20141120
    _effective_regions.b37.bed --add- known-indels b37/1000G_phase1.
    indels.b37.vcf.gz --readgroup-info ID:020,PU:PGM/P1.1.17/
    IonXpress_020,PL:IONTORRENT,SM:NA12878,LB:lib1
```

Compared to the commands shown in supporting material A.1, it takes considerably less lines to build a variant calling benchmarking pipeline with *Vcaller*. To fully appreciate the amount of lines saved, refer to the listing in section A.2.

3.4 SUMMARY

All required inputs were collected following a set of criteria that would meet the needs of this study. These input files were passed to pipelines consisting of multiple short read aligner $\times$ variant caller combinations, as well as a common group of supporting software, built according to the guidelines proposed by Van der Auwera et al. (2013). These pipelines were implemented into *Vcaller*, a CLI that automates the variant calling process, allowing the user to easily build their own pipelines. Finally, the *Vcaller* subcommands used to pass the Ion Proton GIAB sample through a `BWA-MEM + BCFTools` pipeline were shown (Listing 3.1) to illustrate *Vcaller*'s ability to reduce the number of lines required to perform variant calling.

4

RESULTS AND DISCUSSION

Previously, variant calling pipelines featuring several read aligner and variant caller combinations were implemented and run on exome reads pertaining to the same well-characterized sample and sequenced from multiple platforms belonging to two technologies, Ion Torrent and Illumina. Subsequently, the resulting variant calls were compared against a high-confidence call set to evaluate their performance. In the present chapter, results derived from this variant calling benchmarking process will be examined to determine the strengths and weaknesses of each tool.

The final results for this work include prediction counts (TPs, FPs, and FNs) and performance metrics (precision, recall, and F1 scores) output by *qfy.py*, which have been stratified by variant type (Tables 12 and 13). Through these tables, tools were assessed based on their F1 scores, which, as seen in Section 2.7.2, incorporate both precision and recall, thus serving as a quick way to evaluate individual tool performance.

Precision and recall values for different thresholds of the chosen scoring field (QUAL) were passed to a modified version of the script *rocplot.Rscript* (`https://github.com/Illumina/hap.py/blob/master/src/R/rocplot.Rscript`). The resulting PR curves, also stratified by variant type, are shown in Fig. 7 and Fig. 8, and offer a visual way of comparing performance across different tools.

4.1 VARIANT REPRESENTATION DIFFERENCES MATTER IN CALL COMPARISON

Because *vcfeval* normalizes variant representation to ensure that matching variants are considered the same irrespective of differences in their VCF representation, the choice of either benchmark or query representation for tallying TPs will influence their counts. For example, in the case of a MNP affecting three neighboring nucleotides, the benchmark may opt to break it into three SNP calls (three TPs) while the query may consider the MNP as a unit (one TP). Therefore, *vcfeval* opts to write two VCF files containing the matching variants, one using the benchmark call set's representation, and the other that of the query call set. The *hap.py* script accounts for this scenario, and thus produces counts for both representations, which can sometimes be quite different. Furthermore, as previously dis-

cussed, FPs are variants present in the query call set but absent from the benchmark call set, and therefore can only be counted relative to the query set's representation. Likewise, FNs are variants found in the benchmark call set but not the in query call set, and so have to be counted from the benchmark set's VCF. Consequently, precision is calculated through query TPs and FPs, and recall using benchmark TPs and FNs (refer to equations 4 and 5) to preserve consistency.

4.2 F1 SCORES HELP EVALUATE THE STRENGTHS AND WEAKNESSES OF EACH TOOL

Tables 12 and 13 contain all counts and prediction metrics output by *qfy.py* for each pipeline, stratified by variant type. In the two following subsections, tools will be evaluated based on their F1 scores for SNP and indel detection.

4.2.1 *SNP subset*

The results in Table 12 appear to be independent of read aligner, since most of the time the best variant caller is the same irrespective of the choice of aligner. If the TMAP + TVC pipeline is ignored (because it is optimized for Ion Torrent data sets), BCFTools consistently offers the best F1 scores for Ion Torrent data sets. As for Illumina data sets, both BCFTools and Varscan 2 provide good results. In the specific case of the Illumina HiSeq2500 data set, performance is similar for most pipelines, with the exception of non-BCFTools variant callers paired with TMap, which perform much worse. Notably, VarScan 2 pipelines show very good results when calling variants on the Illumina TruSeq data set, with F1 values very close to those of BCFTools.

4.2.2 *Indel subset*

According to the results in Table 13, the F1 scores for indel detection in Ion Torrent data sets were very low across the board. Further, as previously seen, read aligners barely exerted an influence on variant calling, although it is noticeable that TMap did not perform well when integrating pipelines for the Illumina HiSeq250 data set, unless its variants were called with BCFTools.

Again, the best result for Ion Torrent was Ion Proton GIAB's TMAP + TVC pipeline; other than that, VarScan 2 consistently provided the best indel detection results for Ion Torrent data sets. Illumina data sets, on the other hand, are generally capable of reaching F1 scores much closer to those obtained in SNP detection, with BCFtools and HaplotypeCaller having the overall best performances. Similarly to the SNP subset, calling variants on

Table 12: Confusion matrix counts and related performance measures for the SNP variant type.

			B.TP	B.FN	Q.TP	Q.FP	Recall	Precision	F1 Score
Ion Proton GIAB	Bowtie 2	BCFTools	24259	8187	24355	1804	0.748	0.931	0.829
		Freebayes	26117	6329	25935	10833	0.805	0.705	0.752
		HaplotypeCaller	22028	10418	22086	2315	0.679	0.905	0.776
		VarScan 2	18856	13590	18930	2964	0.581	0.865	0.695
	bwa	BCFTools	26311	6135	26419	2155	0.811	0.925	0.864
		Freebayes	26709	5737	26509	6226	0.823	0.810	0.816
		HaplotypeCaller	23643	8803	23714	2423	0.729	0.907	0.808
		VarScan 2	19599	12847	19676	2736	0.604	0.878	0.716
	TMap	BCFTools	23880	8566	23976	1989	0.736	0.923	0.819
		Freebayes	25043	7403	24878	3747	0.772	0.869	0.818
		HaplotypeCaller	22358	10088	22423	2378	0.689	0.904	0.782
		VarScan 2	16938	15508	17003	2943	0.522	0.852	0.648
		Torrent Variant Caller	31169	1277	31110	262	0.961	0.992	**0.976**
Ion Proton Brescia	Bowtie 2	BCFTools	31034	1412	31168	1160	0.956	0.964	0.960
		Freebayes	31716	730	31445	9324	0.978	0.771	0.862
		HaplotypeCaller	30080	2366	30193	5314	0.927	0.850	0.887
		VarScan 2	30206	2240	30339	2097	0.931	0.935	0.933
	bwa	BCFTools	32058	388	32205	1581	0.988	0.953	**0.970**
		Freebayes	32082	364	31798	7981	0.989	0.799	0.884
		HaplotypeCaller	31196	1250	31314	7262	0.961	0.812	0.880
		VarScan 2	31584	862	31716	1591	0.973	0.952	0.963
	TMap	BCFTools	31916	530	32058	1611	0.984	0.952	0.968
		Freebayes	32023	423	31748	8637	0.987	0.786	0.875
		HaplotypeCaller	19989	12457	20063	7878	0.616	0.718	0.663
		VarScan 2	31296	1150	31433	1945	0.965	0.942	0.953
Illumina TruSeq	Bowtie 2	BCFTools	26249	1611	26404	2610	0.942	0.910	0.926
		Freebayes	26432	1428	26232	14331	0.949	0.647	0.769
		HaplotypeCaller	26120	1740	26267	4126	0.938	0.864	0.899
		VarScan 2	25257	2603	25407	804	0.907	0.969	0.937
	bwa	BCFTools	26304	1556	26460	2911	0.944	0.901	0.922
		Freebayes	26458	1402	26252	15098	0.950	0.635	0.761
		HaplotypeCaller	26273	1587	26423	4508	0.943	0.854	0.896
		VarScan 2	25242	2618	25392	826	0.906	0.968	0.936
	TMap	BCFTools	26094	1766	26246	1752	0.937	0.937	**0.937**
		Freebayes	26388	1472	26189	8894	0.947	0.746	0.835
		HaplotypeCaller	26194	1666	26344	2628	0.940	0.909	0.924
		VarScan 2	25208	2652	25357	734	0.905	0.972	**0.937**
Illumina HiSeq2500	Bowtie 2	BCFTools	40304	2069	40618	843	0.951	0.980	**0.965**
		Freebayes	40307	2066	40010	2542	0.951	0.940	0.946
		HaplotypeCaller	40123	2250	40430	826	0.947	0.980	0.963
		VarScan 2	32846	9527	33075	911	0.775	0.973	0.863
	bwa	BCFTools	40028	2345	40331	1382	0.945	0.967	0.956
		Freebayes	40276	2097	39986	5244	0.951	0.884	0.916
		HaplotypeCaller	40140	2233	40458	1138	0.947	0.973	0.960
		VarScan 2	31584	862	31716	1591	0.973	0.952	0.963
	TMap	BCFTools	40015	2358	40313	914	0.944	0.978	0.961
		Freebayes	18345	24028	18186	750	0.433	0.960	0.597
		HaplotypeCaller	18736	23637	18816	346	0.442	0.982	0.610
		VarScan 2	17647	24726	17714	293	0.416	0.984	0.585

B.TP Benchmark TPs **B.FN** Benchmark FNs **Q.TP** Query TPs **Q.FP** Query FPs

Table 13: Prediction counts and related performance metrics for the INDEL variant type.

			B.TP	B.FN	Q.TP	Q.FP	Recall	Precision	F1 Score
Ion Proton GIAB	Bowtie 2	BCFTools	603	946	613	15866	0.389	0.037	0.068
		Freebayes	894	655	916	82637	0.577	0.011	0.022
		HaplotypeCaller	704	845	729	23467	0.454	0.030	0.057
		VarScan 2	622	927	633	11530	0.402	0.052	0.092
	bwa	BCFTools	628	921	642	30485	0.405	0.021	0.039
		Freebayes	920	629	942	128482	0.594	0.007	0.014
		HaplotypeCaller	753	796	777	32603	0.486	0.023	0.044
		VarScan 2	654	895	667	13497	0.422	0.047	0.085
	TMap	BCFTools	709	840	723	12369	0.458	0.055	0.099
		Freebayes	810	739	828	72782	0.523	0.011	0.022
		HaplotypeCaller	682	867	702	25286	0.440	0.027	0.051
		VarScan 2	521	1028	531	8146	0.336	0.061	0.104
		TVC	1049	500	1072	4988	0.677	0.177	**0.281**
Ion Proton Brescia	Bowtie 2	BCFTools	1175	374	1203	47794	0.759	0.025	0.048
		Freebayes	1235	314	1266	134854	0.797	0.009	0.018
		HaplotypeCaller	1094	455	1139	91738	0.706	0.012	0.024
		VarScan 2	1065	484	1086	43011	0.688	0.025	0.048
	bwa	BCFTools	1240	309	1267	50447	0.801	0.025	0.048
		Freebayes	1276	273	1306	176456	0.824	0.007	0.015
		HaplotypeCaller	1162	387	1210	126775	0.750	0.009	0.019
		VarScan 2	1130	419	1157	42472	0.730	0.027	0.051
	TMap	BCFTools	1185	364	1212	27533	0.765	0.042	**0.080**
		Freebayes	1233	316	1259	137174	0.796	0.009	0.018
		HaplotypeCaller	509	1040	528	103252	0.329	0.005	0.010
		VarScan 2	1086	463	1106	41312	0.701	0.026	0.050
Illumina TruSeq	Bowtie 2	BCFTools	1070	487	1094	446	0.687	0.710	0.699
		Freebayes	1279	278	1316	2179	0.821	0.377	0.516
		HaplotypeCaller	1253	304	1307	695	0.805	0.653	0.721
		VarScan 2	1164	393	1188	419	0.748	0.739	0.743
	bwa	BCFTools	1057	500	1083	619	0.679	0.636	0.657
		Freebayes	1292	265	1327	2217	0.830	0.374	0.516
		HaplotypeCaller	1288	269	1349	688	0.827	0.662	0.736
		VarScan 2	1133	424	1160	415	0.728	0.737	0.732
	TMap	BCFTools	1153	404	1178	294	0.741	0.800	**0.769**
		Freebayes	1268	289	1302	1574	0.814	0.453	0.582
		HaplotypeCaller	1264	293	1324	647	0.812	0.672	0.735
		VarScan 2	1098	459	1123	399	0.705	0.738	0.721
Illumina HiSeq2500	Bowtie 2	BCFTools	2789	1157	2859	796	0.707	0.782	0.743
		Freebayes	3242	704	3304	2465	0.822	0.573	0.675
		HaplotypeCaller	3285	661	3405	1470	0.832	0.698	**0.760**
		VarScan 2	2290	1656	2330	522	0.580	0.817	0.679
	bwa	BCFTools	2833	1113	2897	811	0.718	0.781	0.748
		Freebayes	3183	763	3245	2807	0.807	0.536	0.644
		HaplotypeCaller	3286	660	3412	1544	0.833	0.688	0.754
		VarScan 2	2460	1486	2510	582	0.623	0.812	0.705
	TMap	BCFTools	2653	1293	2709	415	0.672	0.867	0.757
		Freebayes	457	3489	465	213	0.116	0.686	0.198
		HaplotypeCaller	441	3505	457	82	0.112	0.848	0.197
		VarScan 2	368	3578	374	61	0.093	0.860	0.168

B.TP Benchmark TPs **B.FN** Benchmark FNs **Q.TP** Query TPs **Q.FP** Query FPs

Illumina HiSeq2500 aligned with TMap leads to very poor F1 scores, unless it is paired with BCFTools.

4.3 Precision-Recall curves provide a visual way to compare variant calling performance

By themselves, performance metrics cannot provide an overview of performance for varying scoring field thresholds; rather, one must resort to threshold-free measures such as *Receiver Operating Characteristic (ROC)* or PR curves. But which to choose?

In most biological problems, the negative subset far outweighs the positive subset (Berrar and Flach, 2012). Data sets derived from variant analysis suffer from this imbalance issue, given that there is an overwhelming number of TNs, i.e. loci where no deviation from the reference is observed. For such binary classifiers—either the locus has a variant or it does not—ROC curves fail to illustrate the true face of the data, since they are built using *True Positive Rate (TPR)* (the same as recall, see Equation 5) and *False Positive Rate (FPR)* (also known as specificity), which is defined as:

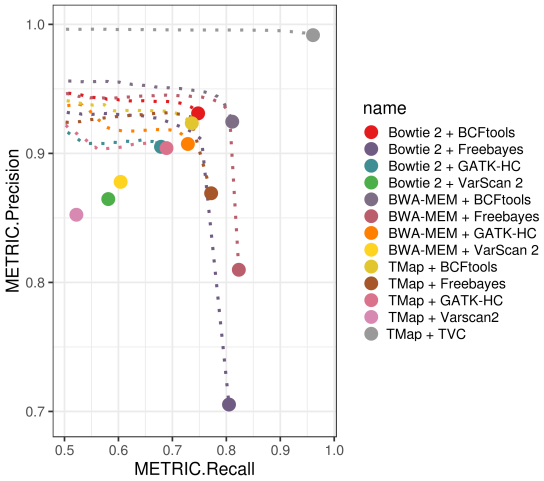
$$\frac{FalsePositives}{TrueNegatives + FalsePositives} \tag{7}$$

Because the number of TNs is so high, the number of FPs in Equation 7 becomes irrelevant; thus, FPR is practically a constant in the context of variant detection, and therefore useless as an illustration of variation in threshold-free measures.

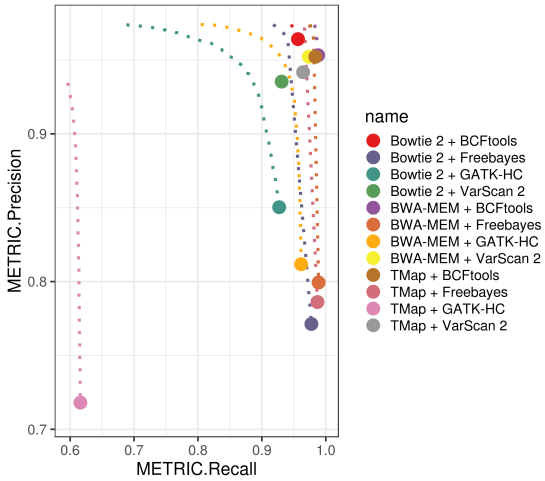
On the other hand, as their name implies, PR curves leverage precision and recall (Equations 4 and 5). Because TNs are not a part of precision and recall definitions, PR curves make it easier to visually appreciate changes in the case of imbalanced sets in which negative counts far outweigh positive ones (Davis and Goadrich, 2006; Saito and Rehmsmeier, 2015). For this reason, this work will forgo the use of ROC curves, and instead focus on exploring PR curves.

The plots in Fig. 7 and Fig. 8 show how precision and recall vary with the scoring threshold for each pipeline; the closer a PR curve gets to the upper-right-hand corner, the better.

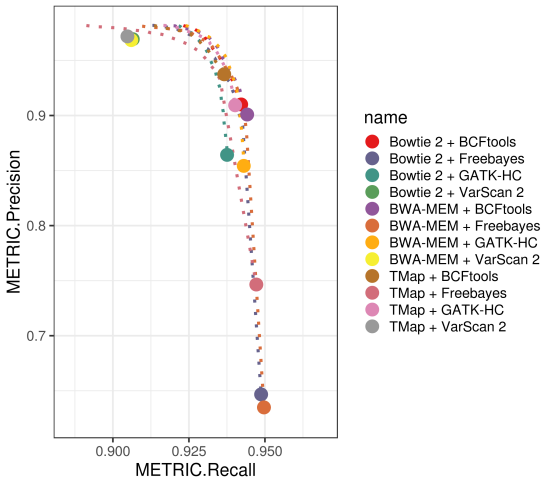
Because pipelines featuring VarScan 2 had a constant value of zero for the QUAL field, their PR curves could not be plotted.



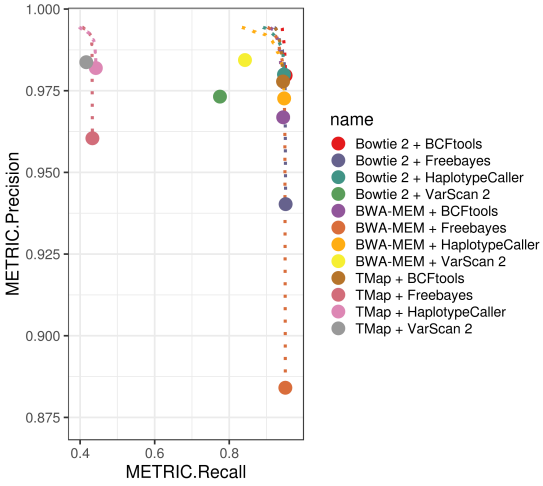
(a) Ion Proton GIAB SNPs



(b) Ion Proton Brescia SNPs

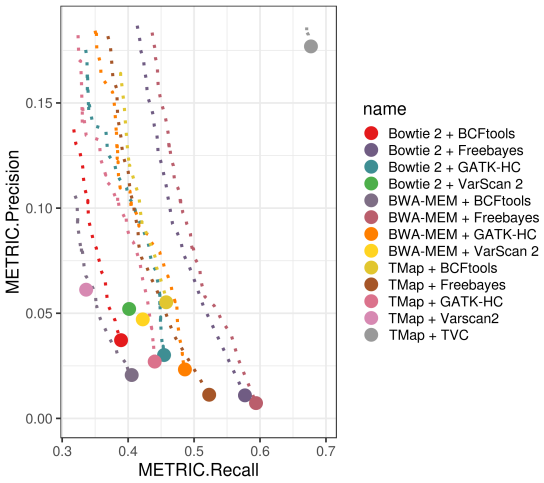


(c) Illumina TruSeqSNPs

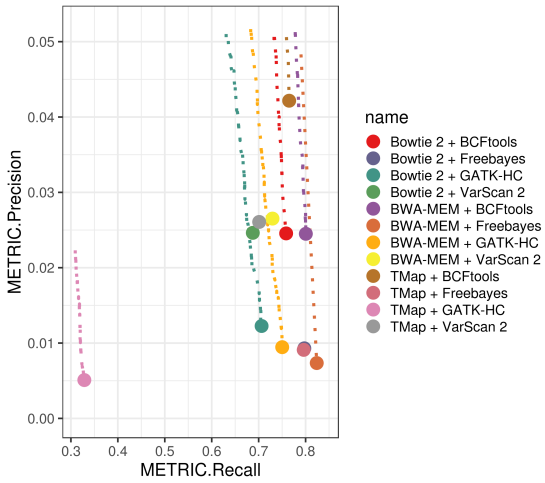


(d) Illumina HiSeq2500 SNPs

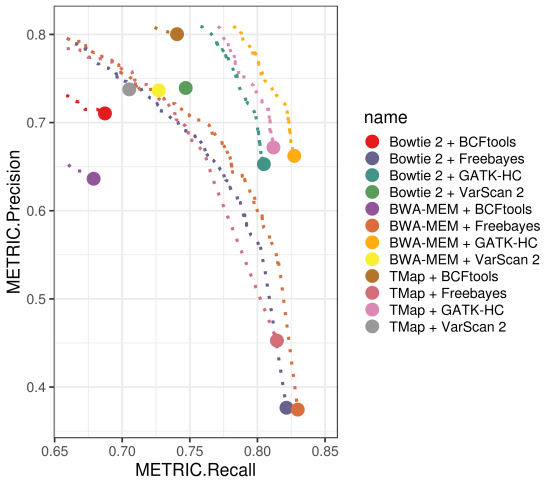
Figure 7: PR curves for the SNP subset in the four data sets under study. Independently of sequencing platform, pipelines featuring Freebayes have low precision, especially when paired with Bowtie 2. For Ion Torrent, BCFTools and HaplotypeCaller (GATK-HC) cluster together at good levels of recall and precision, with the notable exception of the TMap + HaplotypeCaller pipeline on the Ion Proton Brescia data set. Another notable outlier, TMap + TVC, is found in the Ion Proton GIAB data set, with undoubtedly the best result for this technology. Illumina data sets have overall solid performance; the only other notable underperformers are most pipelines featuring TMap in the Illumina HiSeq2500 data sets.



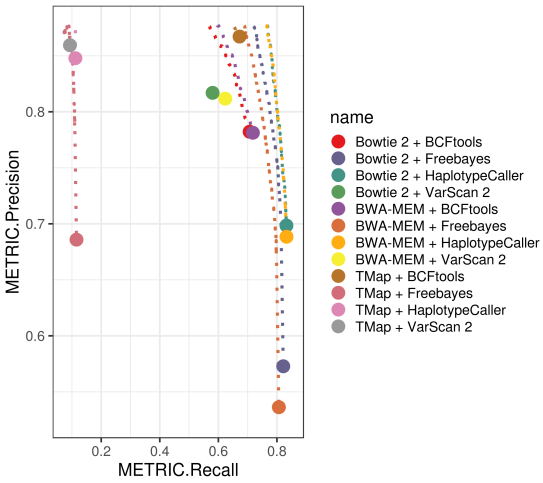
(a) Ion Proton GIAB Indels



(b) Ion Proton Brescia Indels



(c) Illumina TruSeq Indels



(d) Illumina HiSeq2500 Indels

Figure 8: PR curves for the INDEL subset in the four data sets under study. Similarly to SNP analysis, pipelines featuring Freebayes have low precision, especially in the case of Ion Torrent data sets; this might be due to the fact that indel detection precision is overall very low for those data sets. The TMap + TVC and TMap + HaplotypeCaller (GATK-HC) pipelines are conspicuous outliers for the Ion Proton GIAB and Ion Proton Brescia data sets, respectively. Lastly, the combination of TMap with any variant caller that is not BCFTools results in very low levels of recall in the Illumina HiSeq2500 data set.

4.3.1 *Ion Proton GIAB*

SNPs

The PR curves for pipelines featuring Freebayes reach good levels of recall for SNP detection, but only at the cost of a high degree of precision, especially in the case of the Bowtie 2 + HaplotypeCaller pipeline (Fig. 7a). On the other hand, VarScan 2's results have solid levels of precision, but its levels of recall are the lowest among all results for this data set.

Irrespective of read aligner, the pipelines featuring HaplotypeCaller and BCFTools appear clustered together, with good precision (>0.90), although they are only able to reach average levels of recall. As expected, the TMap + TVC pipeline produces the best curve, capable of reaching the optimal upper-right-hand corner.

From these results, short read aligners appear to make a modest contribution to variant calling performance, with curves clustering mostly based on variant calling method.

Indels

From the range of precision it is noticeable that, no matter how optimal a curve may be, indel detection precision will always be very poor (Fig. 8a).

The best result for this data set belongs, as expected, to the TMap + TVC pipeline. The remaining pipelines show even lower levels of precision—less than 0.10—with only the pipelines featuring Freebayes being able to find more than half the indels present in the benchmark call set, albeit at the cost of having a very high fraction of FPs—over 98% of all of its indel calls.

4.3.2 *Ion Proton Brescia*

SNPs

For SNPs, there are three well-defined clusters (Fig. 7b). The first is made up only of the TMap + HaplotypeCaller pipeline, and offers the worst performance from among all SNP detection results for this data set. Then, there is a cluster of high-recall low-precision results encompassing all pipelines featuring Freebayes, alongside the BWA-MEM + HaplotypeCaller and Bowtie 2 + HaplotypeCaller pipelines. Lastly, there is a cluster with high levels of both precision and recall, where all pipelines featuring BCFTools and VarScan 2 are found.

Again, short read aligners appear to have little to no influence on variant calling performance.

Indels

As with Ion Proton GIAB, the range of precision for Ion Proton Brescia's indels is limited to very low precision values, in this case at or below 0.05 (Fig. 8b). Conversely, all pipelines, with the exception of TMap + HaplotypeCaller, exhibit above average recall (>0.70).

Freebayes-based pipelines have the worst precision, as do the two remaining HaplotypeCaller PR curves, Bowtie 2 + HaplotypeCaller and BWA-MEM + HaplotypeCaller. Pipelines that use BCFTools or VarScan 2 stand at a slightly better precision for similar levels of recall, with the TMap + BCFTools pipeline distinguishing itself as the best curve in this plot.

4.3.3 *Illumina TruSeq*

SNPs

In this data set, all pipelines featuring VarScan 2 are closely clustered, with very high precision and recall for SNPs (Fig. 7c). Although no pipeline is present at the upper-right-hand corner of the plot, it is important to note that the range of recall varies only from a little under 0.90 to just below 1. For their part, pipelines using Freebayes continue displaying high recall but low precision.

Pipelines featuring BCFTools and HaplotypeCaller are clustered together, and offer the best balance of precision and recall for this data set's SNPs.

Indels

Unlike Ion Torrent data sets, Illumina data sets proved capable of reaching indel detection precision of upwards to 0.90 at best, or close to 0.30 at worst (Fig. 8c).

Freebayes-based pipelines again exhibit high levels of recall but low precision. On the other end of the spectrum are the BCFTools and VarScan 2 pipelines, which show higher levels of precision but lower recall—although the trade-off is never as steep. The TMap + BCFTools pipeline stands out for offering the highest precision (∼0.80), while still possessing close to 0.75 recall.

Pipelines featuring HaplotypeCaller are clustered together, and offer a balance of decent precision (>0.65) and good recall (>0.80).

4.3.4 *Illumina HiSeq2500*

SNPs

For the first time in this study, a cluster was formed around the curves of pipelines featuring the same read aligner, TMap, rather than the same variant caller (Fig. 7d). These pipelines exhibit high SNP detection precision, but also very poor recall.

Once again, pipelines using Freebayes possess lower levels of precision, although, since the range of precision only varies between 0.875 and 1, the difference is not as glaring as in previous plots, meaning that, with the exception of the TMap + Freebayes pipeline, Freebayes pipelines offer decent results.

BCFTools, HaplotypeCaller, and VarScan 2—with the exception of TMap-based pipelines in the case of the latter two—all show good results, as their curves draw close to the upper-right-hand corner of the plot.

Indels

Similarly to what was observed in the SNP subset for this data set, there is clustering around one of the read aligners, TMap (Fig. 8d). The PR curves for this cluster range from average to high precision, but suffer from low recall ($<$0.20). The exception to this tendency is the TMap + BCFTools pipeline, which does in fact offer the highest level of precision for this data set ($>$0.85), while maintaining acceptable levels of recall ($\sim$0.70).

Pipelines featuring VarScan 2 and BCFTools that were not aligned using TMap offer a decent balance of precision ($>$0.75) and recall ($>$0.50), with slightly superior results in the case of the latter variant caller.

5

CONCLUSION

5.1 CONCLUSIONS

Variant calling results appear to be independent of the choice of short read aligner. The exception to the rule is TMap, which performs particularly poorly on the data sets sequenced on more than one lane, Illumina HiSeq2500 and Ion Proton Brescia. Nevertheless, there is one TMap pipeline, TMAP + BCFTOOLS, which performs similarly to, or at times even better than, most other pipelines, across all data sets.

As for variant callers, both BCFTools and HaplotypeCaller perform well overall, although the latter has some issues, such as poor performance when operating on reads aligned by TMap. Freebayes's results were lackluster, with its only saving grace being its high recall values, which were nonetheless comparable to those of other pipelines with much better precision. Conversely, VarScan 2 tended to have high precision but lower recall, especially in the case of indels, and performed particularly poorly on the Ion Proton GIAB data set.

The TMAP + TVC pipeline performs much better than any other for Ion Proton GIAB, which might stem from the fact that these two tools were developed for joint use on Ion Torrent sequences. Moreover, the GIAB Consortium ran this pipeline on this exact same data set to integrate the resulting calls into their high-confidence data set, leading to a potential bias in the benchmark call set when comparing it against this specific pipeline and data set combination.

From these results, we conclude that, with regard to Ion Torrent data sets, the TMAP + TVC pipeline should be used if flow information is available; else, BCFTools together with any read aligner is a solid option as well. For this latter variant caller, performance is slightly improved for SNP detection if BWA-MEM is used to align the reads, and indel detection is enhanced by aligning reads with TMap.

Concerning Illumina data sets, the best results were observed when BCFTools was combined with any short read aligner, or VarScan 2 combined with any aligner excluding TMap (due to the very low recall obtained in the Illumina HiSeq2500 data set). Overall, BCFTools offers a solid variant calling performance for all Illumina data sets irrespective of the choice of read aligner, with TMap offering a potential for slightly higher indel detection.

When stratifying result comparison by technology, we find that, as far as Ion Torrent platforms are concerned, Ion Proton GIAB is at an advantage due to being able to leverage the TMap + TVC pipeline, which has the best results for either of the two Ion Proton data sets; as previously discussed that pipeline is, however, very likely prone to bias. As far as other pipelines are concerned, performance was very similar for SNP and indel calling across the two data sets.

In the case of Illumina platforms, Illumina TruSeq exhibits a significantly narrower and higher range of recall for both SNPs and indels. On the other hand, Illumina HiSeq250 has a narrower and higher range of precision. As far as the range of recall is concerned, however, the brunt of Illumina HiSeq2500's issues stem from the result of the TMap cluster. Considering the range of precision, TruSeq's precision problems appear to arise because of the results for Freebayes, although this issue is observed across all data sets to some extent.

Overall, SNP detection performance is good for both technologies averaging between 87.4% for Illumina and 84.4% for Ion Torrent. BCFtools pipelines offer the best, or runner-up results for both technologies, with VarScan 2 also performing similarly well on Illumina data sets. On the other hand, Ion Torrent indel detection performance is very poor, with an average F1 score of 5.6% on Ion and a much better F1 score of 63.1% on Illumina. The low precision levels reflected on the F1 scores reveals that there is still much work to be done in terms of improving detection of indels, a type of variant largely disregard in the clinical practice.

5.2 LIMITATIONS

As is always the case with this type of study, the results and any conclusions drawn from them cannot be taken at face value. When speaking of a "best" pipeline, it might simply be a tendency observed in the benchmarked data sets, because results require a large amount of data points if they are to be robust. Even then, if only sequences obtained from one sample are analyzed, results might be prone to be more descriptive of the sample itself rather than, for instance, the level of variant calling (WGS, WES, gene panels, etc.) under study. For these reasons, there are likewise several aspects of this work that may have contributed to the presence of bias in the results.

First, all benchmarking was conducted on sequences obtained from a single sample, NA12878. More importantly, only data sets sequenced from that sample were analyzed, making the results prone to sample-specific biases. This issue was further compounded when comparing technologies, an important component of this study, because each technology (Ion Torrent and Illumina) was limited to two data sets. Moreover, as previously discussed, only one of the two Ion Torrent data sets possessed the read information required to run the `TMap + TVC` pipeline, and this data set integrated the high-confidence call

set used in benchmarking, leading to a very high potential for bias for that pipeline. Lastly, tests were performed on human genome build b37, which is not based on the most recent human genome assembly—GRCh38.

Moreover, it is important to note that VarScan 2 deviated from its default settings by having a few parameters set to custom values in order to approximate its results to those of HaplotypeCaller and BCFTools (refer to the commands in supporting material A.1). Freebayes could likely benefit from such tweaking as well, although the optimization of tool parameters is beyond the scope of this work.

5.3 Prospect for future work

The primary objective of this work has been fulfilled. Nonetheless, there could have been more emphasis on the clinical applications, which would benefit from analyzing what impact different tool parameters have on variant calling performance. Although variant calling parameter optimization is an exciting prospect, it would be very complex to explore in practice, however, because different methods have distinct sets of parameters that cannot be directly compared. Regardless, this type of analysis could lead to large improvements in the field, and is therefore a topic worth considering for future research.

To overcome the limitations inherent to the present work, it would be beneficial to benchmark more samples, and, more importantly, a larger number of data sets sequenced from those samples using each technology, making it possible to draw more robust and informative conclusions. Particularly, a greater number of Ion Torrent data sets containing flow information would greatly aid in the assessment of the performance of the Ion Torrent-specific `TMap + TVC` pipeline.

To expand on this work, it would be interesting for future studies to be performed on human genome assembly GRCh38, especially as GIAB works on improving its GRCh38 benchmark call sets, further leading to more supporting materials for that assembly becoming available. Moreover, the analysis could be extended to WGS as the latter becomes more affordable and timely, or even restricted to gene sequencing panels, for example; ultimately the goal of variant calling benchmarking is to determine what pipelines are better suited to a given variant detection task, and therefore the scope of the analysis ought to be chosen primarily to suit the needs of researchers.

As for technologies, results for indel detection are very unsatisfactory for Ion Torrent platforms. This could be due to two important characteristics of the data generated by these platforms: reads are of variable length, and, unlike Illumina, its principal error mode is associated with miscalling of homopolymer lengths, which undoubtedly has a negative impact on indel detection. Therefore, one can postulate that algorithms still have room for improvement when it comes to proper treatment of Ion Torrent data. To expand on this

point, in the future more technologies and/or sequencing platforms could be benchmarked to help drive algorithm development and improve performance of both new and existing tools when dealing with non-Illumina data.

BIBLIOGRAPHY

A. Auton et al. A global reference for human genetic variation. *Nature*, 526(7571):68–74, Oct 2015.

M. J. Bamshad, S. B. Ng, A. W. Bigham, H. K. Tabor, M. J. Emond, D. A. Nickerson, and J. Shendure. Exome sequencing as a tool for Mendelian disease gene discovery. *Nat. Rev. Genet.*, 12(11):745–755, Sep 2011.

R. Bao, L. Huang, J. Andrade, W. Tan, W. A. Kibbe, H. Jiang, and G. Feng. Review of current methods, applications, and data management for the bioinformatics analysis of whole exome sequencing. *Cancer Inform*, 13(Suppl 2):67–82, 2014.

D. Berrar and P. Flach. Caveats and pitfalls of ROC analysis in clinical microarray research (and how to avoid them). *Brief. Bioinformatics*, 13(1):83–97, Jan 2012.

L. M. Bragg, G. Stone, M. K. Butler, P. Hugenholtz, and G. W. Tyson. Shining a light on dark sequencing: characterising errors in Ion Torrent PGM data. *PLoS Comput. Biol.*, 9(4): e1003031, Apr 2013.

M. Burrows and D. J. Wheeler. A block-sorting lossless data compression algorithm. Technical Report 124, Digital Equipment Corporation, 1994.

J. G. Cleary, R. Braithwaite, K. Gaastra, B. S. Hilbush, S. Inglis, S. A. Irvine, A. Jackson, R. Littin, M. Rathod, D. Ware, J. M. Zook, L. Trigg, and F. M. De La Vega. Comparing variant call files for performance benchmarking of next-generation sequencing variant calling pipelines. *bioRxiv*, 2015. doi: 10.1101/023754. URL https://www.biorxiv.org/content/early/2015/08/03/023754.

A. Cornish and C. Guda. A Comparison of Variant Calling Pipelines Using Genome in a Bottle as a Reference. *Biomed Res Int*, 2015:456479, 2015.

J. Davis and M. Goadrich. The relationship between precision-recall and roc curves. *Proceedings of the 23rd International Conference on Machine Learning, ACM*, 06, Jun 2006.

M. A. DePristo, E. Banks, R. Poplin, K. V. Garimella, J. R. Maguire, C. Hartl, A. A. Philippakis, G. del Angel, M. A. Rivas, M. Hanna, A. McKenna, T. J. Fennell, A. M. Kernytsky, A. Y. Sivachenko, K. Cibulskis, S. B. Gabriel, D. Altshuler, and M. J. Daly. A framework for variation discovery and genotyping using next-generation DNA sequencing data. *Nat. Genet.*, 43(5):491–498, May 2011.

M. A. Eberle, E. Fritzilas, P. Krusche, M. Kallberg, B. L. Moore, M. A. Bekritsky, Z. Iqbal, H. Y. Chuang, S. J. Humphray, A. L. Halpern, S. Kruglyak, E. H. Margulies, G. McVean, and D. R. Bentley. A reference data set of 5.4 million phased human variants validated by genetic inheritance from sequencing a three-generation 17-member pedigree. *Genome Res.*, 27(1):157–164, 01 2017.

A. K. Emde, M. H. Schulz, D. Weese, R. Sun, M. Vingron, V. M. Kalscheuer, S. A. Haas, and K. Reinert. Detecting genomic indel variants with exact breakpoints in single- and paired-end sequencing data using SplazerS. *Bioinformatics*, 28(5):619–627, Mar 2012.

P. Ferragina and G. Manzini. Opportunistic data structures with applications. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, pages 390–398. IEEE, 2000.

L. Feuk, A. R. Carson, and S. W. Scherer. Structural variation in the human genome. *Nat. Rev. Genet.*, 7(2):85–97, Feb 2006.

E. Garrison and G. Marth. Haplotype-based variant detection from short-read sequencing. *ArXiv e-prints*, July 2012.

G. S. Ginsburg and H. F. Willard. Genomic and personalized medicine: foundations and applications. *Transl Res*, 154(6):277–287, Dec 2009.

S. Goodwin, J. D. McPherson, and W. R. McCombie. Coming of age: ten years of next-generation sequencing technologies. *Nat. Rev. Genet.*, 17(6):333–351, 05 2016.

J. M. Heather and B. Chain. The sequence of sequencers: The history of sequencing DNA. *Genomics*, 107(1):1–8, Jan 2016.

S. Hwang, E. Kim, I. Lee, and E. M. Marcotte. Systematic comparison of variant calling pipelines using gold standard personal exome variants. *Sci Rep*, 5:17875, Dec 2015.

International Human Genome Sequencing Consortium. Initial sequencing and analysis of the human genome. *Nature*, 409(6822):860–921, Feb 2001.

D. C. Koboldt, Q. Zhang, D. E. Larson, D. Shen, M. D. McLellan, L. Lin, C. A. Miller, E. R. Mardis, L. Ding, and R. K. Wilson. VarScan 2: somatic mutation and copy number alteration discovery in cancer by exome sequencing. *Genome Res.*, 22(3):568–576, Mar 2012.

P. Krusche, L. Trigg, P. C. Boutros, C. E. Mason, F. M. De La Vega, B. L. Moore, M. Gonzalez-Porta, M. A. Eberle, Z. Tezak, S. Labadibi, R. Truty, G. Asimenos, B. Funke, M. Fleharty, M. Salit, J. M. Zook, and Global Alliance for Genomics and Health Benchmarking Team. Best practices for benchmarking germline small variant calls in human genomes. *bioRxiv*,

2018. doi: 10.1101/270157. URL https://www.biorxiv.org/content/early/2018/02/23/270157.

B. Langmead and S. L. Salzberg. Fast gapped-read alignment with Bowtie 2. *Nat. Methods*, 9(4):357–359, Mar 2012.

S. Laurie, M. Fernandez-Callejo, S. Marco-Sola, J. R. Trotta, J. Camps, A. Chacon, A. Espinosa, M. Gut, I. Gut, S. Heath, and S. Beltran. From Wet-Lab to Variations: Concordance and Speed of Bioinformatics Pipelines for Whole Genome and Whole Exome Sequencing. *Hum. Mutat.*, 37(12):1263–1271, 12 2016.

W. P. Lee, M. P. Stromberg, A. Ward, C. Stewart, E. P. Garrison, and G. T. Marth. MOSAIK: a hash-based algorithm for accurate next-generation sequencing short-read mapping. *PLoS ONE*, 9(3):e90581, 2014.

H. Li. A statistical framework for SNP calling, mutation discovery, association mapping and population genetical parameter estimation from sequencing data. *Bioinformatics*, 27 (21):2987–2993, Nov 2011.

H. Li and R. Durbin. Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics*, 25(14):1754–1760, Jul 2009.

H. Li and N. Homer. A survey of sequence alignment algorithms for next-generation sequencing. *Brief. Bioinformatics*, 11(5):473–483, Sep 2010.

H. Li, B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis, and R. Durbin. The Sequence Alignment/Map format and SAMtools. *Bioinformatics*, 25(16): 2078–2079, Aug 2009a.

R. Li, Y. Li, X. Fang, H. Yang, J. Wang, K. Kristiansen, and J. Wang. SNP detection for massively parallel whole-genome resequencing. *Genome Res.*, 19(6):1124–1132, Jun 2009b.

X. Liu, S. Han, Z. Wang, J. Gelernter, and B. Z. Yang. Variant callers for next-generation sequencing data: a comparison study. *PLoS ONE*, 8(9):e75619, 2013.

E. R. Mardis. The impact of next-generation sequencing technology on genetics. *Trends Genet.*, 24(3):133–141, Mar 2008.

A. McKenna, M. Hanna, E. Banks, A. Sivachenko, K. Cibulskis, A. Kernytsky, K. Garimella, D. Altshuler, S. Gabriel, M. Daly, and M. A. DePristo. The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. *Genome Res.*, 20(9):1297–1303, Sep 2010.

M. L. Metzker. Sequencing technologies - the next generation. *Nat. Rev. Genet.*, 11(1):31–46, Jan 2010.

D. Muzzey, E. A. Evans, and C. Lieber. Understanding the Basics of NGS: From Mechanism to Variant Calling. *Curr Genet Med Rep*, 3(4):158–165, 2015.

S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, 48(3):443–453, Mar 1970.

R. Nielsen, J. S. Paul, A. Albrechtsen, and Y. S. Song. Genotype and SNP calling from next-generation sequencing data. *Nat. Rev. Genet.*, 12(6):443–451, Jun 2011.

J. O'Rawe, T. Jiang, G. Sun, Y. Wu, W. Wang, J. Hu, P. Bodily, L. Tian, H. Hakonarson, W. E. Johnson, Z. Wei, K. Wang, and G. J. Lyon. Low concordance of multiple variant-calling pipelines: practical implications for exome and genome sequencing. *Genome Med*, 5(3):28, 2013.

S. Pabinger, A. Dander, M. Fischer, R. Snajder, M. Sperk, M. Efremova, B. Krabichler, M. R. Speicher, J. Zschocke, and Z. Trajanoski. A survey of tools for variant analysis of next-generation genome sequencing data. *Brief. Bioinformatics*, 15(2):256–278, Mar 2014.

R. Parikh, A. Mathai, S. Parikh, G. Chandra Sekhar, and R. Thomas. Understanding and using sensitivity, specificity and predictive values. *Indian J Ophthalmol*, 56(1):45–50, 2008.

M. Pirooznia, M. Kramer, J. Parla, F. S. Goes, J. B. Potash, W. R. McCombie, and P. P. Zandi. Validation and assessment of variant calling pipelines for next-generation sequencing. *Hum. Genomics*, 8:14, Jul 2014.

R. Poplin, V. Ruano-Rubio, M. A. DePristo, T. J. Fennell, M. O. Carneiro, G. A. Van der Auwera, D. E. Kling, L. D. Gauthier, A. Levy-Moonshine, D. Roazen, K. Shakir, J. Thibault, S. Chandran, C. Whelan, M. Lek, S. Gabriel, M. J. Daly, B. Neale, D. G. MacArthur, and E. Banks. Scaling accurate genetic variant discovery to tens of thousands of samples. *bioRxiv*, 2018. doi: 10.1101/201178. URL https://www.biorxiv.org/content/early/2018/07/24/201178.

R. Redon, S. Ishikawa, K. R. Fitch, L. Feuk, G. H. Perry, T. D. Andrews, H. Fiegler, M. H. Shapero, A. R. Carson, W. Chen, E. K. Cho, S. Dallaire, J. L. Freeman, J. R. Gonzalez, M. Gratacos, J. Huang, D. Kalaitzopoulos, D. Komura, J. R. MacDonald, C. R. Marshall, R. Mei, L. Montgomery, K. Nishimura, K. Okamura, F. Shen, M. J. Somerville, J. Tchinda, A. Valsesia, C. Woodwark, F. Yang, J. Zhang, T. Zerjal, J. Zhang, L. Armengol, D. F. Conrad, X. Estivill, C. Tyler-Smith, N. P. Carter, H. Aburatani, C. Lee, K. W. Jones, S. W. Scherer, and M. E. Hurles. Global variation in copy number in the human genome. *Nature*, 444(7118):444–454, Nov 2006.

K. Reinert, B. Langmead, D. Weese, and D. J. Evers. Alignment of Next-Generation Sequencing Reads. *Annu Rev Genomics Hum Genet*, 16:133–151, 2015.

T. Saito and M. Rehmsmeier. The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PLoS ONE*, 10(3): e0118432, 2015.

S. T. Sherry, M. H. Ward, M. Kholodov, J. Baker, L. Phan, E. M. Smigielski, and K. Sirotkin. dbSNP: the NCBI database of genetic variation. *Nucleic Acids Res.*, 29(1):308–311, Jan 2001.

T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *J. Mol. Biol.*, 147(1):195–197, Mar 1981.

A. Talwalkar, J. Liptrap, J. Newcomb, C. Hartl, J. Terhorst, K. Curtis, M. Bresler, Y. S. Song, M. I. Jordan, and D. Patterson. SMaSH: a benchmarking toolkit for human genome variant calling. *Bioinformatics*, 30(19):2787–2795, Oct 2014.

L. Tattini, R. D'Aurizio, and A. Magi. Detection of Genomic Structural Variants from Next-Generation Sequencing Data. *Front Bioeng Biotechnol*, 3:92, 2015.

G. A. Van der Auwera, M. O. Carneiro, C. Hartl, R. Poplin, G. Del Angel, A. Levy-Moonshine, T. Jordan, K. Shakir, D. Roazen, J. Thibault, E. Banks, K. V. Garimella, D. Altshuler, S. Gabriel, and M. A. DePristo. From FastQ data to high confidence variant calls: the Genome Analysis Toolkit best practices pipeline. *Curr Protoc Bioinformatics*, 43:1–33, 2013.

A. Warr, C. Robert, D. Hume, A. Archibald, N. Deeb, and M. Watson. Exome Sequencing: Current and Future Perspectives. *G3 (Bethesda)*, 5(8):1543–1550, Jul 2015.

E. A. Worthey, A. N. Mayer, G. D. Syverson, D. Helbling, B. B. Bonacci, B. Decker, J. M. Serpe, T. Dasu, M. R. Tschannen, R. L. Veith, M. J. Basehore, U. Broeckel, A. Tomita-Mitchell, M. J. Arca, J. T. Casper, D. A. Margolis, D. P. Bick, M. J. Hessner, J. M. Routes, J. W. Verbsky, H. J. Jacob, and D. P. Dimmock. Making a definitive diagnosis: successful clinical application of whole exome sequencing in a child with intractable inflammatory bowel disease. *Genet. Med.*, 13(3):255–262, Mar 2011.

X. Yu and S. Sun. Comparing a few SNP calling algorithms using low-coverage sequencing data. *BMC Bioinformatics*, 14:274, Sep 2013.

J. Zook, J. McDaniel, H. Parikh, H. Heaton, S. A. Irvine, L. Trigg, R. Truty, C. Y. McLean, F. M. De La Vega, C. Xiao, S. Sherry, and Genome in a Bottle Consortium Salit, M. and. Reproducible integration of multiple sequencing datasets to form high-confidence snp, indel, and reference calls for five human genome reference materials. *bioRxiv*, 2018. doi: 10.1101/281006. URL https://www.biorxiv.org/content/early/2018/05/25/281006.

J. M. Zook, B. Chapman, J. Wang, D. Mittelman, O. Hofmann, W. Hide, and M. Salit. Integrating human sequence data sets provides a resource of benchmark SNP and indel genotype calls. *Nat. Biotechnol.*, 32(3):246–251, Mar 2014.

J. M. Zook, D. Catoe, J. McDaniel, L. Vang, N. Spies, A. Sidow, Z. Weng, Y. Liu, C. E. Mason, N. Alexander, E. Henaff, A. B. McIntyre, D. Chandramohan, F. Chen, E. Jaeger, A. Moshrefi, K. Pham, W. Stedman, T. Liang, M. Saghbini, Z. Dzakula, A. Hastie, H. Cao, G. Deikus, E. Schadt, R. Sebra, A. Bashir, R. M. Truty, C. C. Chang, N. Gulbahce, K. Zhao, S. Ghosh, F. Hyland, Y. Fu, M. Chaisson, C. Xiao, J. Trow, S. T. Sherry, A. W. Zaranek, M. Ball, J. Bobe, P. Estep, G. M. Church, P. Marks, S. Kyriazopoulou-Panagiotopoulou, G. X. Zheng, M. Schnall-Levin, H. S. Ordonez, P. A. Mudivarti, K. Giorda, Y. Sheng, K. B. Rypdal, and M. Salit. Extensive sequencing of seven human genomes to characterize benchmark reference materials. *Sci Data*, 3:160025, Jun 2016.

A

SUPPORTING MATERIAL

A.1 VARIANT CALLING PIPELINES: DETAILED LIST OF COMMANDS

All commands were executed on an environment running Linux Ubuntu 16.04.

A.1.1 *Read alignment*

Because not all tools in this pipeline support zipped FASTA files (e.g. GATK), it was necessary to run the command `gunzip` on the reference genome and associated files:

```
1 $ gunzip b37/human_g1k_v37.*.gz
```

Listing A.1: Uncompressing all zipped files relative to the reference genome.

Due to computational speed concerns, all three read aligners have the indexing of the reference genome FASTA file as a mandatory step, performed as follows:

Bowtie 2

```
1 $ bowtie2-dir/bowtie2-build b37/human_g1k_v37.fasta b37/
    human_g1k_v37
```

BWA-MEM

```
1 $ bwa index b37/human_g1k_v37.fasta
```

TMap

```
1 $ tmap-dir/tmap index -f b37/human_g1k_v37.fasta
```

Afterwards, the commands for read alignment proper were run:

Bowtie 2

```
$ bowtie2-dir/bowtie2 -x b37/human_g1k_v37 -S bt2_output.sam
    query_read [query_read2]
```

BWA-MEM

```
$ bwa mem -M b37/human_g1k_v37.fasta query_read [query_read2] >
   bwa_output.sam
```

TMap

```
$ tmap-dir/tmap map1 -o 2 -f human_g1k_v37.fasta -r query_read [
   query_read2] > tmap_output.bam
```

The optional parameter in BWA-MEM's command, `-M`, ensures that its output is compatible with Picard, a tool integrated into GATK for data processing. The one in TMap's, `-o 2`, simply redefines the output type as BAM instead of *Sequence Alignment/Map (SAM)*, thus saving the need for running `samtools sort` to convert it to BAM like in the case of the two other tools, although in certain cases it might be better to run it anyway as it is often desirable for the reads to be sorted by coordinate, which is necessary for certain downstream steps.

Most reads were aligned separately, including the independently sequenced runs pertaining to the same sample, like the single-end runs of Ion Proton Brescia. The only exception was the only sample in this work with paired-end reads, HiSeq2500 Exome, wherein the `query_read2` positional parameter was defined.

A.1.2 *Post-alignment processing*

Although the b37 resource bundle already contains the fasta file index (faidx) and sequence dictionary for reference genome `human_g1k_v37.fasta`, it is important to note that these two files are fundamental to run tools in certain downstream steps (e.g. GATK). One can generate them using the two following commands:

```
$ samtools faidx human_g1k_v37.fasta
$ gatk4-dir/gatk CreateSequenceDictionary -R b37/human_g1k_v37.
   fasta
```

Listing A.2: How to index and create a sequence dictionary for the reference genome's FASTA sequence.

The aligned reads were converted to the BAM format, and their contents sorted by their coordinate, with subsequent indexing of the resulting BAM file:

```
$ samtools sort -O bam -o aligned_reads.sorted.bam -T /tmp/
    sorted_output.tmp aligned_reads.sam
$ samtools index aligned_reads.sorted.bam
```

Listing A.3: Sorting aligned reads by coordinate, and indexing the resulting BAM file.

The `-T` parameter sets the prefix for any temporary files generated during the sorting process, and, optionally, a target directory wherein to store them.

Another processing step that was necessary for downstream steps was the addition of read group information, which consists of tab-separated metadata concerning the sample under analysis. A unique ID, sample name, library identifier, sequencing platform, and sequencing platform unit (e.g. run barcode) were thus provided:

```
$ gatk4-dir/gatk AddOrReplaceReadGroups \
-I aligned_reads.sorted.bam \
-O aligned_reads.sorted.RG.bam \
-RGID 42 \
-RGSM NA1278 \
-RGPL lib1
-RGPL IONPROTON \
-RGPU unit1
```

Listing A.4: An example of how to add read group information to a set of aligned reads.

PCR amplification ensures that there will be a sufficient amount of DNA to work with in downstream steps, and is therefore required in library preparation. In spite of its usefulness, PCR has the issue of generating duplicate reads that should not be used as evidence for or against potential variants. Likewise, the optical sensors of sequencing machines can create optical duplicates. It is therefore advisable to mark and remove these duplicate reads (also known as "dedupping"), which was accomplished with the following command:

```
$ gatk4-dir/gatk MarkDuplicates \
-I aligned_reads.sorted.RG.bam \
-O aligned_reads.sorted.RG.dedupped.bam \
--REMOVE_DUPLICATES True \
-M aligned_reads.sorted.metrics
```

Listing A.5: Marking and removing duplicate reads from the sorted BAM.

The `-M` parameter, while not required for the purposes of this work, is required by the above command; it defines the name of the file in which to write duplication metrics for analysis. Meanwhile, the optional flag `--REMOVE_DUPLICATES` ensures that, instead of being written again with the appropriated flags set, i.e., merely marked, the duplicates are truly removed.

Short read aligners might align reads on the edges of indels, creating mismatches between bases, which can lead to the incorrect calling of SNPs. Realignment around known indels is a two-step process that aims to address this issue. First, a list of intervals in need of intervention is generated, and that list is then passed to a command able to perform the realignment proper:

```
$ java -jar gatk3-dir/GenomeAnalysisTK.jar \
-T RealignerTargetCreator \
-I aligned_reads.sorted.dedupped.bam \
-R b37/human_g1k_v37.fasta \
-o aligned_reads.sorted.dedupped.intervals \
-known b37/1000G_phase1.indels.b37.vcf.gz \
-known b37/Mills_and_1000G_gold_standard.indels.b37.vcf.gz
```

Listing A.6: Generation of the intervals to undergo realignment.

```
$ java -jar gatk3-dir/GenomeAnalysisTK.jar \
-T IndelRealigner \
-I aligned_reads.sorted.dedupped.bam \
-R b37/human_g1k_v37.fasta \
-targetIntervals aligned_reads.sorted.dedupped.intervals \
-o aligned_reads.sorted.dedupped.realigned.bam \
-known b37/1000G_phase1.indels.b37.vcf.gz \
-known b37/Mills_and_1000G_gold_standard.indels.b37.vcf.gz
```

Listing A.7: Realigning around indels within the generated intervals.

The `-known` (one dash) parameter takes as input a VCF file containing a list of known indel sites. Although it can only take one input file, the option can be repeated as many times as necessary, as shown in the above example commands. This is yet another way in which the GATK Resource Bundle proves useful, since it contains two sets of known indel sites suited for the chosen reference human genome build: Mills indels and 1000 Genomes indels.

NGS platforms produce reads in the FASTQ format, which combines FASTA sequences with their associated per-base Phred-scaled base quality scores. Base quality scores express

a sequencer's confidence in its calling the correct base at each position, consequently making it possible to evaluate associated sequencing machine errors. The emission of base quality scores can be subject to systematic errors, however, which can negatively impact their usefulness. Thus, a machine learning approach capable of detecting systematic errors so as to model them and readjust base quality scores, BQSR, was run on the data. Note that the machine learning algorithm requires at least one database of known polymorphic sites, such as dbSNP (Sherry et al., 2001), to build its model.

```
$ gatk4-dir/gatk BaseRecalibrator \
-I aligned_reads.sorted.dedupped.realigned.bam \
-R b37/human_g1k_v37.fasta \
-o aligned_reads.sorted.dedupped.realigned.table \
-knownSites b37/dbsnp_138.b37.vcf.gz
```

Listing A.8: Creating a covariance table for a sample using a list of known sites.

```
$ gatk4-dir/gatk ApplyBQSR \
-I aligned_reads.sorted.dedupped.realigned.bam \
-R b37/human_g1k_v37.fasta \
-o aligned_reads.sorted.dedupped.realigned.bqsr.bam \
-bqsr aligned_reads.sorted.dedupped.realigned.table
```

Listing A.9: Applying base score recalibration based on the covariance table.

A.1.3 *Variant calling*

BCFtools

A component of the Samtools suite, BCFtools calls variants after calculating the genotype likelihoods for the provided aligned reads:

```
$ bcftools mpileup -Ob -o bcf_output.bcf -f b37/human_g1k_v37.
    fasta aligned_reads.sorted.dedupped.realigned.bqsr.bam
$ bcftools call -vm -O v -o raw_calls.vcf aligned_reads.sorted.
    dedupped.realigned.bqsr.bam
```

In the `bcftools mpileup` command, `-Ob` sets the output as being in the *Binary Call Format (bcf)* format. As for the `bcftools call` command, `-O v` sets the final call set output as being in the VCF format, while `-vm` ensures that only variant sites are written to the resulting VCF, and that the multiallelic caller is used in place of the default consensus caller, which is

fundamental for accounting for instances where more than two alleles are being compared—for example, when more than one sample is called at a time.

HaplotypeCaller

GATK's HaplotypeCaller performs reassembly of the aligned reads to build sequences (haplotypes) at variant sites before calculating their respective genotype likelihoods and assigning genotype calls at that site.

```
$ gatk4-dir HaplotypeCaller --dbsnp b37/dbsnp_138.b37.vcf.gz -R b37/human_g1k_v37.fasta -I aligned_reads.sorted.dedupped.realigned.bam -O raw_calls.vcf
```

The optional `--dbsnp` parameter has no bearing on the calculations; rather, it serves to populate the resulting VCF's ID column.

Freebayes

With the support of Bayesian statistics, Freebayes strings short sequences (haplotypes) on which it calls variants, forsaking any additional information provided by the reference genome used in reassembly.

```
$ freebayes-dir/freebayes -f b37/human_g1k_v37.fasta aligned_reads.sorted.dedupped.realigned.bam > raw_calls.vcf
```

Varscan 2

Entries in a mpileup file are parsed to ensure that they meet certain thresholds, and variants called at positions where Fisher's Exact Test p-value is over a minimum value.

```
$ samtools mpileup -f b37/human_g1k_v37.fasta aligned_reads.sorted.dedupped.realigned.bam > pileup_file.mpileup
$ java -jar varscan2-dir/VarScan.jar mpileup2cns pileup_file.mpileup --output-vcf 1 --variants 1 --p-value 0.10 --min-coverage 2 > raw_calls.vcf
```

Setting the parameter `--output-vcf` ensures that the final output is written to a VCF file, and likewise `--variants 1` makes it so that only variant positions will be reported; `--p-value` and `--min-coverage` set a minimum threshold for p-value and number of reads at each position (coverage) for variants to be called.

Torrent Variant Caller

Rather than being restricted own variant calling algorithm, TVC puts together preexisting tools with their own original modules: in general, putative variants are discovered on sites

determined heuristically using flow space information—exclusively emitted by Ion Torrent sequencers—through a combination of results from Freebayes and an indel assembly module.

```
$ tvc-dir/tvc -r b37/human_g1k_v37.fasta -i aligned_reads.sorted.
  dedupped.realigned.bam -o raw_calls.vcf
```

A.1.4 *Variant calling benchmarking*

In preparation for call set comparison with *vcfeval*, the raw variant calls obtained above were intersected with BED files defining the regions captured by their respective exome kits, and indexed by *tabix*, which takes as input tab-delimited genome position files sorted and compressed by the *bgzip* utility. Before that, however, the BED files had to undergo a minor chromosome notation conversion to comply with the b37 build.

```
$ sed 's/chr//g' truseq-exome-targeted-regions-manifest-v1-2.bed >
  truseq-exome-targeted-regions-manifest-v1-2.b37.bed
$ sed 's/chr//g' nexterarapidcapture_expandedexome_targetedregions.
  bed > nexterarapidcapture_expandedexome_targetedregions.b37.bed
$ sed 's/chr//g' AmpliseqExome.20141120_effective_regions.bed >
  AmpliseqExome.20141120_effective_regions.b37.bed
```

Listing A.10: Converting the exome kit region BED files to the b37 chromosome notation.

```
$ bedtools intersect -header -a ion_bwa_bcftools.vcf -b
  AmpliseqExome.20141120_effective_regions.b37.bed >
  ion_exome_bwa_bcftools.vcf
```

Listing A.11: Example of exome intersection using raw variants called by BCFtools on Ion Proton GIAB reads aligned using BWA-MEM.

```
$ bgzip ion_exome_bwa_bcftools.vcf
$ tabix -p vcf ion_exome_bwa_bcftools.vcf.gz
```

Listing A.12: Sorting and compression of the BWA-MEM + BCFtools Ion Proton GIAB exome-only call set, with subsequent indexing of the resulting file.

The Linux version of *vcfeval* was obtained from the RTG website (https://www.realtimegenomics.com/products/rtg-tools). Each of the four samples went through three read aligners ×

four variant callers pipelines, plus the `TMap + TVC` pipeline which was run exclusively on sample Ion Proton GIAB, for a total of 13 distinct pipelines.

RTG's *Sequence Data File (SDF)* format organizes sequence data into multiple binary files within a directory, increasing the processing efficiency of large data sets. Therefore, for the sake of computational gains, *vcfeval* requires that the reference genome be converted into the SDF format before taking it as input.

```
$ rtg-dir/rtg format -o b37/human_g1k_v37.sdf b37/human_g1k_v37.
    fasta
```

Listing A.13: Converting the FASTA reference genome into the SDF format.

In all cases, the comparison inputs consisted of a query call set, its associated exome regions, the benchmark call set, and the benchmark call set's high-confidence regions.

```
$ rtg-dir/rtg vcfeval -o ion-bwa-bcftools --vcf-score-field QUAL,
    --output-mode ga4gh, --template b37/human_g1k_v37.sdf --calls
    ion_exome_bwa_bcftools.vcf.gz --bed-regions AmpliseqExome
    .20141120_effective_regions.b37.bed --baseline giab/
    NA12878_GIAB_highconf_CG-IllFB-IllGATKHC-Ion-Solid-10X_CHROM1-
    X_v3.3_highconf.vcf.gz --evaluation-regions giab/
    NA12878_GIAB_highconf_CG-IllFB-IllGATKHC-Ion-Solid-10X_CHROM1-
    X_v3.3_highconf.bed
```

Listing A.14: Comparing an example query set against the NA12878 benchmark set using *vcfeval*.

Two optional parameters beget explanation: `--vcf-score-field` sets what VCF field to use as the ROC score, and `--output-mode ga4gh` defines the output report mode to be GA4GH-compliant, so that the comparison results may be used as input in the quantification step.

A part of the Haplotype Comparison Tools suite (`https://github.com/Illumina/hap.py`), *qfy.py* is responsible for executing the quantification step of variant benchmarking.

```
$ hap.py-dir/qfy.py --write-vcf --write-counts --type ga4gh --
    reference b37/human_g1k_v37.fasta --adjust-conf-regions giab/
    NA12878_GIAB_highconf_CG-IllFB-IllGATKHC-Ion-Solid-10X_CHROM1-
    X_v3.3_highconf.bed -o ion-bwa-bcftools-counts ion-bwa-bcftools/
    output.vcf.gz
```

Listing A.15: Counting matches and mismatches, and computing statistical metrics in a query GA4GH-intermediate VCF using `qfy.py`.

As with *vcfeval*, the output is a directory. To increase the amount of available information, two flags were switched on: `--write-vcf`, which writes an annotated VCF containing all benchmark matches and mismatches, and `--write-counts`, so that *qfy.py* will write advanced counts and metrics. Moreover, `--type` sets the expected input type—in this case, a GA4GH-intermediate file—and `--adjust-conf-regions` restricts the count space to remain within the GIAB high-confidence regions.

A.2 List of commands to call variants with BWA-MEM + BCFTools

Below are all the commands required to run a variant calling workflow similar to the one found in Listing 3.1 without resorting to *Vcaller*.

```
$ bwa index b37/human_g1k_v37.fasta
$ bwa mem -M b37/human_g1k_v37.fasta query_read [query_read2] >
    bwa_output.sam
$ samtools faidx b37/human_g1k_v37.fasta
$ gatk4-dir/gatk CreateSequenceDictionary -R b37/human_g1k_v37.
    fasta
$ samtools sort -O bam -o aligned_reads.sorted.bam -T /tmp/
    sorted_output.tmp aligned_reads.sam
$ samtools index aligned_reads.sorted.bam
$ gatk4-dir/gatk AddOrReplaceReadGroups \
-I aligned_reads.sorted.bam \
-O aligned_reads.sorted.RG.bam \
-RGID 42 \
-RGSM NA1278 \
-RGPL lib1
-RGPL IONPROTON \
-RGPU unit1
$ gatk4-dir/gatk MarkDuplicates \
-I aligned_reads.sorted.RG.bam \
-O aligned_reads.sorted.RG.dedupped.bam \
--REMOVE_DUPLICATES True \
-M aligned_reads.sorted.metrics
$ java -jar gatk3-dir/GenomeAnalysisTK.jar \
-T RealignerTargetCreator \
-I aligned_reads.sorted.dedupped.bam \
-R b37/human_g1k_v37.fasta \
-o aligned_reads.sorted.dedupped.intervals \
```

```
25 --known b37/1000G_phase1.indels.b37.vcf.gz \
26 --known b37/Mills_and_1000G_gold_standard.indels.b37.vcf.gz
27 $ java -jar gatk3-dir/GenomeAnalysisTK.jar \
28 -T IndelRealigner \
29 -I aligned_reads.sorted.dedupped.bam \
30 -R b37/human_g1k_v37.fasta \
31 -targetIntervals aligned_reads.sorted.dedupped.intervals \
32 -o aligned_reads.sorted.dedupped.realigned.bam \
33 -known b37/1000G_phase1.indels.b37.vcf.gz \
34 -known b37/Mills_and_1000G_gold_standard.indels.b37.vcf.gz
35 $ gatk4-dir/gatk BaseRecalibrator \
36 -I aligned_reads.sorted.dedupped.realigned.bam \
37 -R b37/human_g1k_v37.fasta \
38 -o aligned_reads.sorted.dedupped.realigned.table \
39 -knownSites b37/dbsnp_138.b37.vcf.gz
40 $ gatk4-dir/gatk ApplyBQSR \
41 -I aligned_reads.sorted.dedupped.realigned.bam \
42 -R b37/human_g1k_v37.fasta \
43 -o aligned_reads.sorted.dedupped.realigned.bqsr.bam \
44 -bqsr aligned_reads.sorted.dedupped.realigned.table
45 $ bcftools mpileup -Ob -o bcf_output.bcf -f b37/human_g1k_v37.
     fasta aligned_reads.sorted.dedupped.realigned.bqsr.bam
46 $ bcftools call -vm -O v -o raw_calls.vcf aligned_reads.sorted.
     dedupped.realigned.bqsr.bam
47 $ bedtools intersect -header -a ion_bwa_bcftools.vcf -b
     AmpliseqExome.20141120_effective_regions.b37.bed >
     ion_exome_bwa_bcftools.vcf
48 $ bgzip ion_exome_bwa_bcftools.vcf
49 $ tabix -p vcf ion_exome_bwa_bcftools.vcf.gz
50 $ rtg-dir/rtg format -o b37/human_g1k_v37.sdf b37/human_g1k_v37.
     fasta
51 $ rtg-dir/rtg vcfeval -o ion-bwa-bcftools --vcf-score-field QUAL,
     --output-mode ga4gh, --template b37/human_g1k_v37.sdf --calls
     ion_exome_bwa_bcftools.vcf.gz --bed-regions AmpliseqExome
     .20141120_effective_regions.b37.bed --baseline giab/
     NA12878_GIAB_highconf_CG-IllFB-IllGATKHC-Ion-Solid-10X_CHROM1-
     X_v3.3_highconf.vcf.gz --evaluation-regions giab/
```

```
    NA12878_GIAB_highconf_CG-IllFB-IllGATKHC-Ion-Solid-10X_CHROM1-
    X_v3.3_highconf.bed
52 $ hap.py-dir/qfy.py --write-vcf --write-counts --type ga4gh --
    reference b37/human_g1k_v37.fasta --adjust-conf-regions giab/
    NA12878_GIAB_highconf_CG-IllFB-IllGATKHC-Ion-Solid-10X_CHROM1-
    X_v3.3_highconf.bed -o ion-bwa-bcftools-counts ion-bwa-bcftools/
    output.vcf.gz
```

Listing A.16: A generic `bwa + BCFTools` pipeline for variant calling benchmarking.