Strategies for Optimisation of Variant Prediction in Yeast Genomes

Prithika Sritharan

A thesis submitted to the University of East Anglia for the degree of Doctor of Philosophy

University of East Anglia

Quadram Institute Bioscience

September 2021

Abstract

Linear reference genomes have guided the alignment of short sequence reads prior to variant prediction. This approach is, however, fundamentally limited when studying species with high levels of sequence diversity. Variation graphs can overcome such limitations by incorporating several genomes within a bi-directed reference structure. This dissertation explores methodologies that could be utilised to optimise variant prediction within yeast genomes, particularly *Saccharomyces cerevisiae*.

Variation graphs constructed from NCYC and third-party strains were found to increase the ability of reads to align in comparison to the S288c reference graph and linear reference genome. The novel *FAT-CIGAR* toolkit was developed to obtain exact read alignment information from linear and graph-based mappers, in the form of the FAT-CIGAR string. Sequence identity scores calculated from the FAT-CIGAR string showed that the *vg* variation graph produced a greater proportion of reads with perfect mapping (75.3%) whilst the *SevenBridges* variation graph mapped a higher number of reads with greater identity scores (96.4%). The accuracy of variant calling was compared for four graph genome software, determining that the *SevenBridges* variation graph produced the most accurate variant calls (F1 score = 0.995), with the greatest recall (0.991), followed by *FreeBayes* (F1 score = 0.995). The *vg* software produced the least accurate variant calls (F1 score = 0.972 to 0.986) due to calling a greater number of false positive variants.

The *FAT-CIGAR* toolkit also enabled the identification of a novel method of variant filtration, removing aligned reads likely to lead to false positive variant calls. SNP calls from reads filtered on the FAT-CIGAR string by 10 bases and indel calls from reads filtered on the CIGAR string by 30 bases removed the highest proportions of false positives in real and simulated datasets. Consequently, the use of the *FAT-CIGAR* toolkit as a standard methodology in future genomic analyses is recommended.

Contents

List of Figures

List of Tables

Acknowledgements

This work was supported by the UKRI Biotechnology and Biological Sciences Research Council Norwich Research Park Biosciences Doctoral Training Partnership (NRPDTP) as a CASE Award in collaboration with Eagle Genomics.

I would first like to thank my amazing supervisors, Jo Dicks and Katharina T. Huber, without whom I would not have been able to complete this research. I am extremely grateful for all their invaluable advice and the continuous guidance that they provided throughout my PhD. I felt truly blessed to have such kind and caring supervisors whose support was deeply encouraging, especially during the more stressful times. I would also like to thank Ian Roberts, my previous supervisor, for all the help he provided during the first year of this project. A great thanks to Will Spooner for helping obtain the funding for this project and Eleanor Stanley for her time and the keen interest she showed in this project. Another thanks to Ann-Marie Keane and Hannah Trewby for the great advice and friendship that they provided.

Many thanks to Erik Garrison, Adam Novak, Natasa Bezmarevic, Jonas Andreas Sibbesen and Emily McTavish for helping me with queries regarding technical issues with the software utilised in this project.

I want to express my deepest gratitude to my parents for their unconditional love and for always being there for me. Thanks to my aunts for believing in me and taking care of me. Thanks to my siblings, especially my sisters, Prathana and Prajena, for all the joy and adventures they bring into my life. Last but not the least, I would like to thank my husband, Ajanthan, for being my greatest supporter and for all the sacrifices he's made for me. I am forever thankful for his unfailing support and continuous encouragement throughout this process.

Chapter 1

Introduction

This dissertation aims to optimise the computational predictions of sequence variants in yeast genomes through the adoption of new and existing methodologies. In order to understand how this might be achieved, this chapter introduces the importance of DNA sequencing and discusses the role played by advancements in sequencing technologies in improving our understanding of biology. The methods utilised in the traditional linear variant discovery pipeline for the prediction and functional characterisation of variants within a genome are then outlined. The software utilised to carry out each method and the limitations of these methods are subsequently discussed in further detail. The concept of genome graphs as alternative references to improve read alignment against a reference structure are introduced and explored. The importance of yeasts and the potential advantages of studying variation in yeast genomes are discussed. Finally, the specific aims and objectives set out to be achieved by this project are defined.

1.1 Introduction to DNA Sequencing

The field of genomics has seen exponential growth during recent decades, due to vast improvements in DNA sequencing technologies, allowing for the characterization and study of entire genome sequences. However, it was the invention of next-generation sequencing (NGS) technology, a high-throughput method which enabled rapid massively parallel sequencing at low cost, that truly revolutionized the field. The decreasing cost of DNA sequencing has enabled a wider range of biological questions to be addressed and extended its usage into fields such as medicine for clinical diagnostics.

Some of the earliest methods for DNA sequencing date back to 1977, when the Maxam-Gilbert method and the Sanger sequencing methods were first developed. The Maxam-Gilbert method employed the use of chemical degradation to fragment DNA radioactively labelled with Phosphorus-32. The modified DNA was cleaved with piperidine to be visualised through electrophoresis (Maxam and Gilbert, 1977). This method was soon phased out due to its complexity and replaced by the more refined Sanger sequencing method (Saccone and Pesole, 2003). The Sanger method, developed by Frederick Sanger, provided the backbone for the development of modern sequencing technologies. This method relied on an analogue of the deoxyribonucleotides (dNTPs), known as the dideoxyribonucleotides (ddNTPs), which lacks the 3' hydroxyl group necessary to form a phosphodiester bond with the 5' phosphate of the next dNTP, resulting in immediate chain termination (Chidgeavadze *et al.*, 1984). Radioactively labelled ddNTPs were incorporated at 0.01 fold concentration of the dNTPs allowing for DNA extension whilst producing random length fragments. This process was carried out in four different sequencing reactions and visualised through autoradiography (Sanger *et al.*, 1977). The method was further improved by replacing radiolabelling with a fluorescent dye and using capillary electrophoresis for improved fluorescence detection (Swerdlow and Gesteland, 1990).

1.1.1 Next Generation Sequencing

The Sanger sequencing method allowed for the development of the first generation of DNA sequencers (see Figure 1.1). The first automated sequencer, AB370A, was introduced by Applied Biosystems in 1986. The capillary electrophoresis-based AB370A was able to sequence 96 samples in parallel with a read length of 600 bp. The development of the pyrosequencing method paved the way for next-generation sequencing with the release of the 454 sequencer by 454 Life Sciences in 2005. This sequencer relied on measuring pyrophosphate production using ATP as a substrate in luciferase-mediated conversion of luciferin to oxyluciferin (Liu *et al.*, 2012). The commercial success of the 454 was shortly followed by the release of other sequencers such as the Illumina Genome Analyzer, each competing to increase the data output in a single sequencing run at reduced cost. The Illumina Genome Analyzer was able to increase data output from 1 Gigabase to 1.8 Terabase by 2014 (Illumina, 2016). Pacific Biosciences developed the first of the third generation sequencers by utilising single-molecule real-time (SMRT)

sequencing to remove the requirement for DNA amplification and enable rapid sequencing. Pacific Biosciences also allows for long read sequencing with read length exceeding 10 Kilobases, facilitating improved *de novo* genome assembly and allowing for direct detection of haplotypes (Schadt *et al.*, 2010). This was followed swiftly by the invention of nanopore sequencers such as MinION by Oxford Nanopore. MinION conducts real-time long read sequencing by measuring electrical conductivity as DNA passes through a biological pore. It has high assembly contiguity which facilitates detection of large-scale structural variants (Lu *et al.*, 2016). The MinION has been increasingly used for genomic surveillance of disease outbreaks such as Ebola (Quick *et al.*, 2016) due to its rapidity and portability.

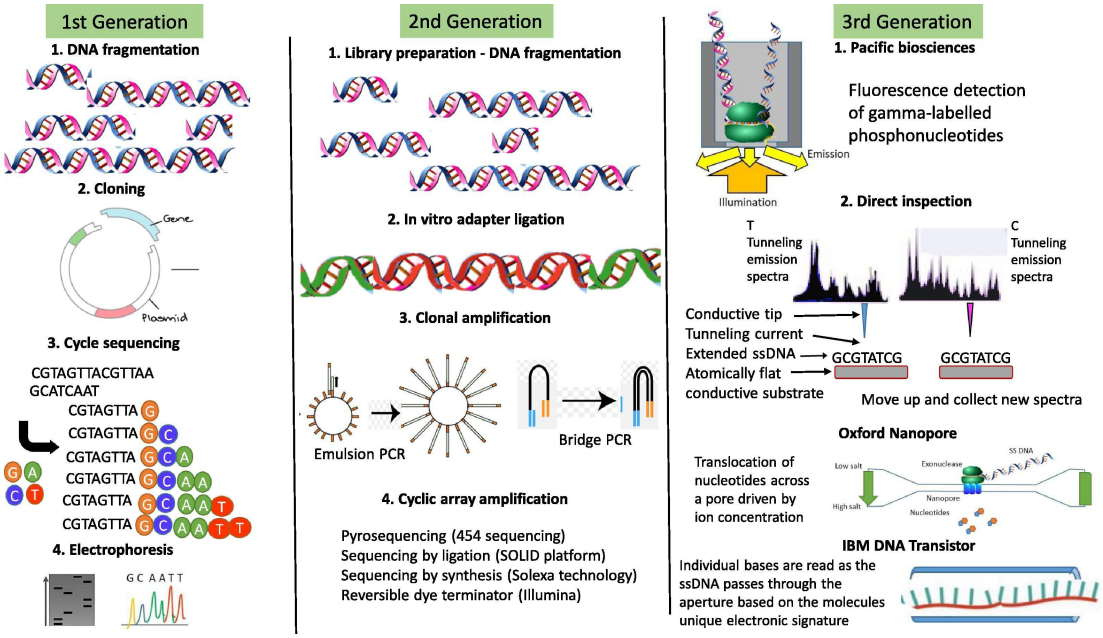


Figure 1.1: **DNA Sequencing Technologies.** This figure highlights the differences in the methods utilised with the advancement in sequencing technologies. The first generation depicts the Sanger sequencing method, the second generation refers to the process of sequencing by synthesis (SBS) used by the majority of sequencers and the third generation sequencers use real-time detection of the genome. This figure was published in Dlamini *et al.*, 2020.

1.1.2 Whole Genome Sequencing

Whole genome sequencing is used to characterize the complete genome sequence of an organism. NGS technology has facilitated the use of whole genome sequencing as a driving force in the research of complex genetic disorders and cancer. The first DNA-based organism to have its complete genome sequenced was the bacteriophage PhiX174, a single-stranded DNA virus, by Fred Sanger (Sanger *et al.*, 1977). This was followed by the whole genome sequencing of the first bacterium, *Haemophilus influenzae* (Fleischmann *et al.*, 1995) and the first archaeon, *Methanocaldococcus jannaschii* (Bult *et al.*, 1996) in 1995. The first eukaryotic organism to be sequenced was *Saccharomyces cerevisiae* in 1996, commonly referred to as baker's yeast, which was determined to have 16 nuclear chromosomes consisting of 12.1 million nucleotides (Goffeau *et al.*, 1996). Other organisms have had their genomes sequenced since due to the increase in large-scale sequencing projects, including the human genome.

The Human Genome Project (HGP) was launched in 1990 to determine the human DNA sequence and achieved this goal by publishing the final draft in 2003, which has since been improved upon, consisting of 2.85 billion nucleotides covering approximately 99% of the euchromatic genome (International Human Genome Sequencing Consortium, 2004).

1.1.3 Population-Level Sequencing

The advance of whole genome sequencing meant that many individuals within a species population could be studied to identify genomic variation such as SNPs (single nucleotide polymorphisms that occur in more than one percent of the population) and indels (small insertions or deletions of bases). Population level sequencing allows for characterization of allelic variants responsible for phenotypic variation and improves understanding of the species diversity and their evolutionary history. Additionally, it can also be used to identify genes that are essential for fitness and survival (Luikart *et al.*, 2003). Many large-scale population level sequencing studies have been carried out in humans such as the 1000 Genomes Project (1KGP) and the International HapMap project. The 1KGP aimed to understand the relationship between genotype and phenotype by extensively cataloging variants through both whole genome and exon-targeted sequencing (1000 Genomes Project Consortium, 2010). The International HapMap

project focused solely on SNPs to build a map of haplotype blocks (a set of SNPs that lie within close proximity on the same chromosome and thus are inherited together) from 269 individuals to identify genetic variants that contribute to disease (The International HapMap Consortium, 2007). Many species of yeasts (*S. cerevisiae* (Liti *et al.*, 2009, Peter *et al.*, 2018), *S. pombe* (Fawcett *et al.*, 2014), *L. kluyveri* (Friedrich *et al.*, 2015)) have also been the centre of population sequencing projects that examine intraspecific genome evolution due to their highly compact genomes (Peter and Schacherer, 2016).

1.1.4 Functional Analyses

The use of whole genome sequencing in large-scale sequencing projects has had a significant impact in enriching our understanding of the function of a genome. Functional analysis of sequenced genomes has allowed for the discovery of novel genes and enabled a better understanding of known genes and their regulatory elements. The increased need to analyse the vast amount of high-throughput data generated from large-scale sequencing experiments has facilitated the initiation of follow-up projects such as the Encyclopedia of DNA elements (ENCODE) and the Genotype-Tissue Expression (GTEx) project. Launched in 2003 after the completion of the HGP, ENCODE aimed to characterize all functional elements in the human genome, with a focus on regions previously determined as "junk" (Encode Project Consortium, 2004). It permitted for biochemical functions to be assigned to over 80% of the human genome and identified SNPs associated with many diseases within the newly characterised non-coding regions (Encode Project Consortium, 2012). The GTEx project conducted a comprehensive study of the variation in gene expression through the characterisation of transcriptomes from 449 human donors. Analysis of the genetic variation within the regulatory networks across multiple tissues revealed several pathways involved in the mechanism of complex diseases (GTEx Consortium, 2017). These studies emphasised the importance of accurate identification and annotation of genetic variations for improved understanding of their role in disease.

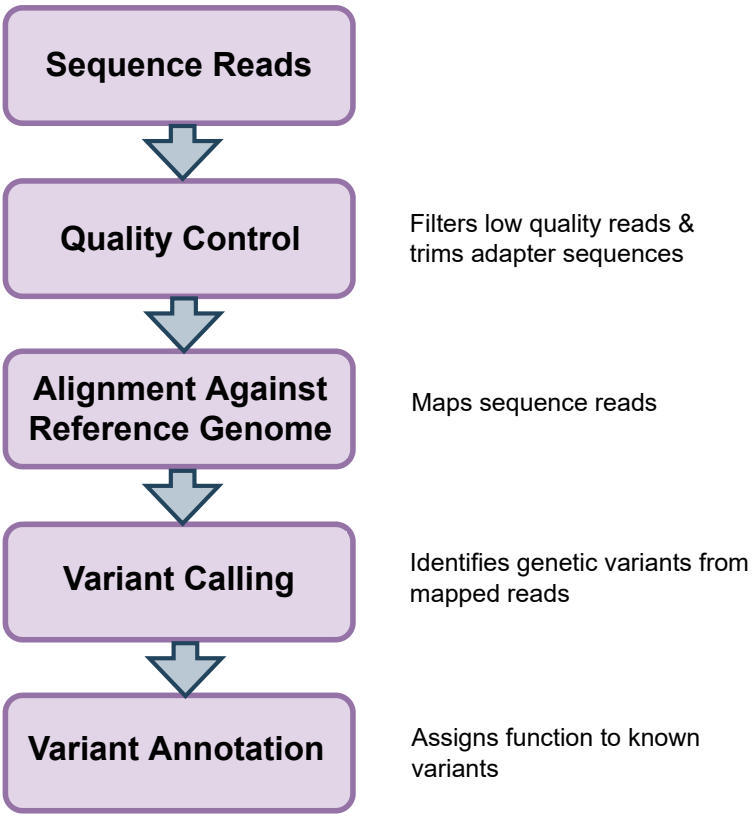


Figure 1.2: **Variant Discovery Pipeline.** This figure is a summary of the methods utilised for the processing of NGS data in order to identify genetic variants. The four main steps in the pipeline are quality control, read mapping, variant calling and variant annotation.

1.2 Traditional Approaches to Variant Discovery

The NGS data analysis pipeline, utilised for the discovery of genetic variants from raw sequence reads, is an established multi-step process that relies on the accuracy of several computational algorithms (see Figure 1.2). Quality control analysis forms the preliminary step of the pipeline and provides an overview of the intrinsic sequence quality. It dictates whether further processing of the raw sequence reads is necessary to improve sequence quality prior to subsequent analysis. The sequence reads are aligned to (mapped) a reference genome, the chosen representative genome containing a set of defining genes for a particular species. In the absence of a reference genome, reads are assembled into contiguous sequence through a *de novo* assembly approach using de Bruijn graphs to represent sequence overlaps (Martin and Wang, 2011). De Bruijn graphs are directed graphs in which nodes are assigned a k-mer sequence of length k which is identical to the adjacent nodes in k-1 bases so that the suffix of one node forms

the prefix of the next (de Bruijn, 1946). Once aligned to a *de novo* or reference genome, any sequences within the mapped reads that differ from the consensus sequence are defined as variants. Variant calling is used to predict true variants by calculating the likelihood of a variant occurring at each individual locus. Hard filtering is the process of filtering variants on one or more specific threshold values. This allows for the removal of false positive variants prior to variant annotation to identify any potential functional effects (Nielsen *et al.*, 2011).

1.2.1 Quality Control

Quality control is of paramount importance to avoid erroneous variant calls that arise from errors within the sequence reads. Sequencing errors may result from a multitude of factors starting with errors in the DNA sequence and library preparation. PCR amplification bias introduces uneven sequence read distribution due to amplification efficiency of certain templates allowing for increased duplication of particular sequence reads (Acinas *et al.*, 2005). The choice of sequencing platform also induces artefacts that are a consequence of technical limitations such as biases in base composition. Illumina flowcells have been found to suppress GC-rich reads during the cluster amplification step (Stein *et al.*, 2010) with the sequencers noted to produce low coverage across GC-rich regions of the genome (Benjamini and Speed, 2012). Many of the sequencers also rely on the process of sequencing by synthesis which leads to the accumulation of errors with each run cycle, causing a drop in the quality of base calling towards the read ends (Fuller *et al.*, 2009). Adapter sequences and read contamination from other samples may also occur during sequencing. These factors lower the overall sequence quality leading to marked impairment in mapping and the misalignment of sequence reads.

There are several software for quality control which utilise the base quality scores output by the sequencer to produce a quality report. The *FastQC* (Andrews, 2010) software has been widely reported in literature (Brown *et al.*, 2017; Li *et al.*, 2017) due to its speed and quality assessment using an extensive range of metrics. *FastQC* provides an overview of the per base sequence quality, average read quality, base composition bias, GC content, presence of duplicated and over-represented sequences and adapter content. These quality metrics are used to inform whether further processing of sequence reads is necessary to improve read quality. Low sequence quality can be

greatly improved by trimming and filtering the reads. Filtering involves removing reads that are extremely short or contain a greater proportion of N's, which indicates that the bases at that position could not be determined. Sequence reads can be trimmed to remove either ends of the reads where there is a reduction in base quality or contamination of adapter sequences. Trimming has been shown to increase read alignment, improve the accuracy of genome assembly and decrease false positive SNP detection in yeasts and other genomes (Del Fabbro *et al.*, 2013).

1.2.2 Read Mapping

The alignment of sequence reads against a reference genome is a crucial step, as all of the downstream analyses including variant calling are highly dependent on its accuracy. The process of read mapping, therefore, needs to be both remarkably precise and extremely fast to be able to handle the millions of reads produced in a typical sequencing run. A wide range of read mapping algorithms have been implemented over the past few years, each one aimed at improving sequence read alignments against a reference genome by catering to a specific biological situation. Most software use a variation of one of the following algorithms: the hashing algorithm or the Burrows-Wheeler Transform (BWT) algorithm (Schbath *et al.*, 2012). Thus selecting the optimal read mapping software to suit the needs of the data also plays an essential role in accurate read analysis, particularly for short sequence reads. Prior knowledge of the genome from which the reads were derived and the level of expected variation with respect to the reference genome may also influence the choice of mapping algorithm. For example, when aligning a highly divergent genome against a reference, it would be advisable to choose a hash-based mapping algorithm like *Stampy* due to its increased sensitivity in mapping (Lunter and Goodson, 2011).

A fundamental challenge for read mapping algorithms is to be able to accurately distinguish between errors caused by sequencing artefacts and true genomic variants within the reads. Departures from the reference genome are primarily defined in alignments by either mismatches or gaps, both of which are considered errors and therefore heavily penalised. Consequently, each mapper takes a different approach to the amount of errors it is willing to tolerate during alignment. Read mappers impose various constraints regarding whether gaps are allowed in alignment, the size of the gaps and the number

of mismatches (Fonseca *et al.*, 2012). These constraints are usually imposed to increase computational efficiency when aligning millions of reads, as performing gapped alignments drastically slows down mapping, particularly for seeding algorithms. Gapped alignments are, however, essential for the accurate variant calling of indels as ungapped alignments can result in reads being unable to align completely or complete misalignment of the reads spanning the long indel regions and the surrounding indel breakpoints. Furthermore, these misaligned reads contain many mismatches that give rise to false positive SNP calls with strong read support (Li and Homer, 2010).

Hash-Based Mapping Algorithms

A hashing algorithm is used for compression of large data to reduce memory requirements. A hash function is utilised by hashing algorithms to convert the input data into a numerical value which is used to index the data within a hash table. For the purpose of read mapping, the hashing algorithm stores a string of sequences as a key which allows for fast look-up of the values (see Figure 1.3). The reference genome is split into k-mers, where the size of k is smaller than the read size and the k-mers are stored as hash keys with the corresponding positions within the genome stored as the value. Sequence reads are then split up into k-mers and mapped against the reference genome by searching against the hash keys. Once the k-mer seeds (small sub-sequences from a read) find possible hits, the seed and extend method is used to map remaining sequences in the read. Seeds are sorted and the compatibility of their positions are assessed for alignments. An advantage of hash-based algorithms is that they are highly sensitive and therefore have reduced mapping bias (Lunter and Goodson, 2011). Limitations of this algorithm are that reads from repeated regions are mapped poorly and it performs slowly as the extension process is time-consuming (Schbath *et al.*, 2012).

Stampy

The *Stampy* (Lunter and Goodson, 2011) software utilises a hash-based algorithm to map short-read sequences to a reference genome. *Stampy* splits the reference genome into sequences of length 15 bases (i.e. k=15) and stores their location in a hash table. In order to speed up the querying of reads, non-unique k-mers that likely arise from repetitive regions are assigned a flag denoting high k-mer count within the hash table.

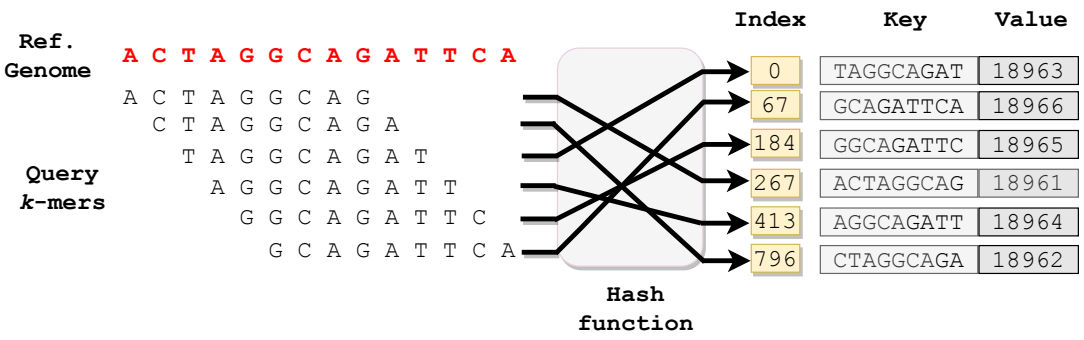


Figure 1.3: **Hash-Based Mapping Algorithm.** This figure is an example of how a hash table is constructed to index the reference genome. K-mer sequences from the reference genome are stored as keys within the hash table and the position of the sequence is stored as the value. During alignment, k-mers from the sequence reads are queried against the hash table to identify possible locations in the reference genome that contains matching query sequences. The sequence reads are aligned by the mapping algorithm against the reference sequence at the retrieved positions in order to determine the optimal alignment.

Equal length k-mers are also constructed from the sequence reads and all overlapping 15-mer sequences across the read are searched against the table to identify potential hits. Sequence similarity filtering is utilised to reduce the number of locations that are targeted as potential candidates. Nucleotide counts are computed for the k-mer and the reference genome sequence at the putative candidate locations. Any differences in the count statistics that exceed a certain threshold results in the filtering of that location.

The read is aligned against the candidate locations using the Needleman-Wunsch dynamic programming algorithm to carry out banded global alignment. The Needleman-Wunsch algorithm was developed to identify the optimal alignment between two sequences. This is computed by initialising a matrix of the sequences to be aligned and determining a scoring system for matches, mismatches and gaps. The top left cell of the matrix is initialised with the score 0 and for each cell, three candidate scores are calculated by considering the score for the cell and the scores from the left, top and the top left diagonal cells. The maximum of the scores is assigned to the cell along with the direction to the cell from which the score was calculated. The path from either the left or top cells represents indels in the alignment whilst the diagonal cell represents either a match or mismatch. The optimal alignment is determined using trace back of the matrix path from the bottom right to the top left cell (Needleman and Wunsch,

1970). A banded alignment places restrictions on gaps with only 15bp indels allowed against the candidate locations. The most likely mapping location is determined by calculating the mapping quality score of the alignment. The read is re-aligned against the best candidate location, allowing for indels of up to length 30bp, to produce the final alignment (Lunter and Goodson, 2011).

The *Stampy* mapping algorithm achieves good sensitivity by utilising a hash table data structure to query k-mers and generating nucleotide count profiles from the read and reference genome to determine candidate locations prior to carrying out global alignment. This process of inexact matching allows for a greater number of reads to be aligned by accounting for the presence of possible variants within the sequence reads. *Stampy* is also unique in that it can be run in hybrid mode to perform alignments using the *BWA* algorithm (Li and Durbin, 2009). This enables majority of reads that have high sequence similarity with the reference genome to be aligned with *BWA* in the first pass prior to utilising *Stampy's* sensitive mapping approach to align reads with lower sequence similarity. The use of the hybrid mode retains sensitivity whilst increasing the run time efficiency (Lunter and Goodson, 2011).

SHRiMP2

Another hash-based read aligner is *SHRiMP2* (Short Read Mapping Programme) (David *et al.*, 2011). *SHRiMP2* also constructs a hash table by indexing the reference genome using multiple spaced seeds instead of consecutive k-mers. Multiple spaced seeds contain spaces within the seed where the algorithm is not concerned about the nucleotide present at that position. The use of spaced k-mer seeds has been previously shown to increase both the sensitivity and speed of mapping (Ilie and Ilie, 2007). The location of each spaced k-mer is stored within the hash table and k-mers with several multiple candidate locations are discarded to prevent exhaustive searching. The spaced seeds from the sequence reads are used to query the hash table. If the same seed is present within the reference sequence, a list of the possible candidate mapping locations is retrieved. *SHRiMP2* employs a paired mapping mode to ensure that reads are only considered for alignment against the candidate mapping location if the location for the read pair also falls within proximal distance to the read. The Smith-Waterman algorithm is employed to align reads against the candidate locations using a two-phased approach. The Smith-

Waterman algorithm is a variation of the Needleman-Wunsch algorithm that performs local instead of global sequence alignments. It does not allow for cells within the matrix to be assigned negative scores, instead these cells are assigned a score of 0. The optimal alignment is identified by tracing back the path from the cell with the highest score until a cell with the score 0 is encountered, thus producing a local alignment (Smith and Waterman, 1981). The first phase only computes the alignment scores for reads against the candidate locations. Candidate locations with the highest alignment scores are re-aligned to obtain the final alignment against the reference genome (David *et al.*, 2011).

BWT-Based Mapping Algorithms

The second major algorithm used by read mappers is the BWT (Burrows and Wheeler, 1994) which is adept at both compression and indexing. This algorithm relies on storing the reference genome in suffix arrays ahead of sorting the suffices and lexicographically ordering them (see Figure 1.4). The suffix array can be traversed using the Ferragina–Manzini (FM) algorithm (Ferragina and Manzini, 2000). For each element in the array, the FM index ranks the occurrence of the element and stores its position. This allows for backward searching by finding the rows containing instances of the first proper suffix from the read in the sorted suffix array and increasing the suffix length until a full match occurs (Langmead, 2013). An advantage of BWT is that it is extremely fast, even for sequence reads that have multiple possible mapping locations, as the repetitive regions are collapsed during array construction (Schbath *et al.*, 2012).

BWA (Burrows-Wheeler Alignment) (Li and Durbin, 2009), a widely reported alignment software (Taliun *et al.*, 2021; Willems *et al.*, 2017; Nurk *et al.*, 2021), implements the BWT algorithm. It has the capacity for fast alignment of both short reads (Li and Durbin, 2009) and long reads (Li and Durbin, 2010). *BWA* creates a prefix tree of the reference genome which is transformed using BWT into a sorted suffix array. The position of each occurrence of the prefix of a read will occur at intervals within the suffix array. Therefore, the suffix array intervals are used to obtain the candidate mapping locations in the genome. Backward searching is used to query the read sequence against suffix array intervals of substrings of the reference genome (Li and Durbin, 2009). The *bwa mem* algorithm, which is used to align short sequence reads, carries out local align-

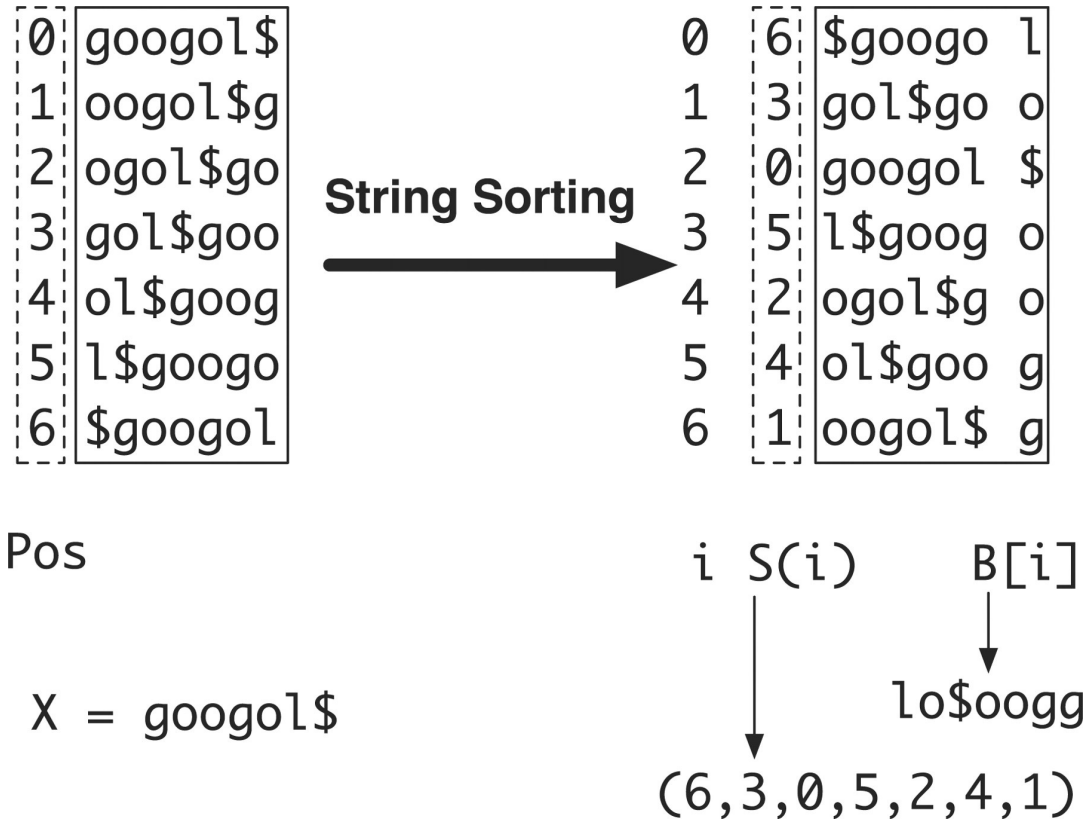


Figure 1.4: **Burrows-Wheeler Transform.** This figure demonstrates how the suffix array is constructed from the string X by lexicographically sorting the suffices. The BWT string is represented by B[i] and S(i) represents the positions stored in the suffix array. This figure was published in Li and Durbin, 2009.

ment using the Smith-Waterman algorithm based on the seed and extend method. Seeds that have super-maximal exact matches (SMEMs), exact matches between the read and reference that cannot be further extended and which do not match to any other position, are identified within the sequence read and co-linear seeds are chained together. Each seed is ranked on the chain and seed length. Local alignments are carried out from the top ranked seeds and the seed is extended if it leads to a potential alignment (Li, 2013). The final alignment is obtained by carrying out banded global alignment of the sequence read with the most probable candidate location determined by the alignment score. *BWA* is extremely fast and has reduced memory usage as only small portions of the array are stored at a time (Li and Durbin, 2009). However, *BWA* slows down significantly if too many errors are allowed as it is not adept at handling errors (Schbath *et al.*, 2012).

Limitations of Mapping

The software explored above differ in speed, accuracy, precision and sensitivity, all of which are important factors to consider in read mapping. However, all of these software share a major limitation which significantly impacts mapping: they align against a linear reference genome. The monoploid reference genome provides a poor representation of the species as a whole. It is the genome of one, or a few depending on the species, individual and therefore only contains a single haplotype at each locus. This poses significant issues as there are many genetic variations naturally present within each individual in a species population, which the reference genome fails to represent. On the contrary, a monoploid reference genome may also carry rare variants that are uncommon to the species resulting in major alleles within the species population remaining unidentified as variant calling is greatly biased towards the reference genome (Audano *et al.*, 2019). Additionally, the absence of variants introduces reference allele bias into mapping. This results in the tendency to over-report alleles present in the reference genome and under-report variant alleles, especially within highly polymorphic regions such as the HLA region. Reference bias can have marked impact on RNA sequencing studies in particular where quantifying allele-specific expression is of greater importance (Degner *et al.*, 2009). Hypervariable regions of genomes that display a high level of sequence variation are unable to be mapped correctly or mapped at all as the reads differ vastly from the reference genome, a problem more prevalent in species such as yeast (Garrison *et al.*, 2018). Similarly, sequence reads that originate from regions of large-scale structural variation, such as insertions, that are not present within the reference genome are also unable to align preventing the discovery of novel genomic variants that may be of functional consequence (Li *et al.*, 2010). A combination of read misassembly and the decline in mapping consequently negatively affects all further downstream analyses. This leads to a number of variants going undetected or called incorrectly, hindering the accurate inference of variants (Paten *et al.*, 2017). It also results in the repeated calling of many common variants within the species as *de novo* variants, limiting the identification of true, novel alleles (Novak *et al.*, 2017).

In an attempt to mitigate this issue, alternative scaffolds were introduced for the hypervariable regions in the release of the latest human genome assembly, GRCh38 (Schneider *et al.*, 2017). GRCh38 contains 178 regions consisting of 261 variant sequences and was identified to have included over 150 genes not present in the primary

assembly. However, these loci are still not used in mapping as current mapping software are not able to handle alternate scaffolds. These scaffolds are treated as paralogous duplications within the genome resulting in the reads being penalized for mapping to multiple locations (Church *et al.*, 2015). This problem further emphasised the necessity for a more sophisticated reference structure and alignment tools that can take known variants into consideration.

1.2.3 Variant Calling

The earliest methods employed by variant calling algorithms simply involved detecting variants directly from alignments by identifying mismatches against the reference genome. The alignments were filtered to keep only high-confidence bases based on a cut-off Phred quality score (a score assigned to each nucleotide base that provides a probability estimate that the called base is incorrect) of Q20, which corresponded to the probability of a base call error of 1% (Ewing *et al.*, 1998). In addition to their cut-off values, frequency filters were then applied based on allele count observations, using the frequency of non-reference bases to infer a genotype. However, this method had several limitations with a major factor being that it did not account for noise errors within the sequence reads. The accuracy of the variant calls were highly dependant on having high coverage sequencing depth as low depth coverage resulted in under-reporting of heterozygous genotypes. In addition, filtering on only quality scores led to loss of information of individual read quality and it did not provide a measure of confidence for the inferred genotypes (Nielsen *et al.*, 2011).

Current variant callers rely on the use of a probabilistic method which estimates the posterior genotype probability within a Bayesian framework whilst taking into account noise and sequencing errors within the read data. The genotype likelihood is obtained for all possible genotypes and is used in combination with the prior genotype probability to identify the maximum posterior genotype probability. This is then used to infer the genotype at each locus. The genotype likelihood, the probability of a genotype being the true genotype given the observed data, is calculated as the product of the probability of the data given the allele frequency and the prior genotype estimation over the sum of all the possible allele frequencies. The read data for the genotype likelihood can be based on the base quality scores (BQ) or mapping quality scores (MAQ), the

probability that the base or read is mapped incorrectly (Li *et al.*, 2008). The genotype likelihood is calculated under the assumption that the reads are independent of one another and therefore relies on the accuracy of alignments and quality score calculations (Li, 2011). The prior genotype probability can also be estimated from available data for the genome. Allele frequencies are obtained from either the reference genome or single nucleotide polymorphism (SNP) databases and the prior probabilities are calculated for each locus assuming the Hardy-Weinberg equilibrium (Neilsen *et al.*, 2012). A constant prior probability can also be assumed for each genotype if the allele frequencies cannot be estimated. As the posterior genotype probability measures the evidence of variation at a site against the probability of there being no variation, it provides a measure of confidence in the variant called. A few of the currently available software for variant calling are outlined below.

GATK HaplotypeCaller

The *Genome Analysis Toolkit (GATK) HaplotypeCaller* (Poplin *et al.*, 2017) is a commonly encountered variant caller in literature (Peter *et al.*, 2018; Priestley *et al.*, 2019; Wang *et al.*, 2020). Traditional pileup variant callers, such as SAMtools (Li *et al.*, 2009), rely on read support to make variant calls and therefore, are able to make precise SNP calls but fail in accuracy when calling small insertions and deletions (indels) due to their dependence on individual alignments. In order to improve variant calling accuracy, the HaplotypeCaller algorithm identifies variant target regions and carries out local *de novo* re-assembly of reads discarding any prior mapping information (see Figure 1.5). Alignment information is utilised to identify target ActiveRegions based on the presence of mismatches, insertions, deletions and soft-clips. The genotype likelihood is estimated for all candidate haplotypes using the pair Hidden Markov Model (pair-HMM) (Durbin *et al.*, 1998), a variation of the HMM that traverses between different states to generate a sequence alignment between the read and candidate haplotypes. The pair-HMM algorithm has three states: match, insertion and deletion. The transition probability from a match to an insertion or deletion state is given a gap opening penalty based on the recalibrated base quality scores. The probability that the alignment remains in the state of either insertion or deletion is given a gap extension penalty of 10. The probability that the state remains a match is a complement of the base error probability given by the base quality. The alignment score calculated from the pair-HMM alignments is

used to calculate the genotype likelihood for each candidate haplotype (Poplin *et al.*, 2017).

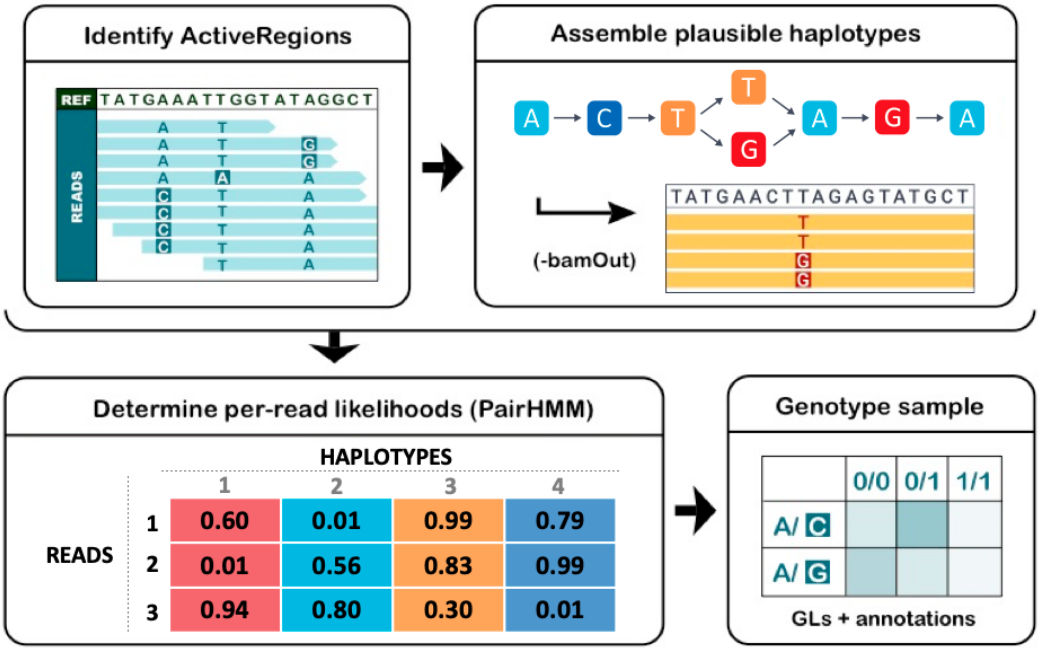


Figure 1.5: **Variant Calling with *GATK HaplotypeCaller*.** This figure shows how target regions are selected and local realignment is carried out to identify candidate haplotypes. The genotype likelihood for candidate haplotypes is estimated using a pair-HMM and the maximum posterior probability is used to determine the genotype. This figure was published in Poplin *et al.*, 2017.

Once the ActiveRegions containing candidate haplotypes have been determined, de Bruijn like graphs are constructed from the reference and read sequence within each ActiveRegion. Each read k-mer is compared against the graph and new nodes are added for every mismatch encountered. Any overlapping reference k-mers are connected into a single path within the graph and the edges are weighted according to the read support for the connecting k-mers. Any edges without sufficient read support, greater than two reads, are pruned out of the graph removing the haplotypes. Candidate haplotypes are re-aligned to the reference using the Smith-Waterman algorithm (Smith and Waterman, 1981), to produce a CIGAR string (see Section 3.2.1) output of the haplotypes which is used to infer the genotype. The genotype likelihood (G_l) is calculated for a diploid genotype composed of two alleles, A_1 and A_2, for a read R at site i, R_i, as the product

over all the reads of the mean likelihood for the alleles in the specified genotype.

$$P(R_i|G_l) = \prod_i \left(\frac{P(R_i|A_1)}{2} + \frac{P(R_i|A_2)}{2} \right) \tag{1.1}$$

The posterior probability, $P(G_l|R_i)$, is evaluated for the candidate haplotypes as the product of the prior probability, $P(G)$, and genotype likelihoods, $P(R_i|G_l)$, over the sum of the genotype likelihood of all the candidate haplotypes.

$$P(G_l|R_i) = \frac{P(G)P(R_i|G_l)}{\sum_l P(R_i|G_l)P(G_l)} \tag{1.2}$$

The haplotype with the maximum posterior probability is assigned as the genotype (Poplin *et al.*, 2017).

The assembly-based variant calling algorithm employed by *GATK HaplotypeCaller* has been shown to improve genotyping accuracy even in low coverage sequencing data for SNP calls (Pirooznia *et al.*, 2014) and was found to have a higher sensitivity for detecting small indels in comparison to other variant callers (Kim *et al.*, 2017). However, a major limitation of the *GATK HaplotypeCaller* is its computational intensity, as increasing the number of samples can result in exponential increases in graph complexity limiting its usability in large-scale sequencing studies (Poplin *et al.*, 2017).

FreeBayes

FreeBayes (Garrison and Marth, 2012) is a haplotype-based variant caller which detects haplotypes from the read sequence itself rather than from read alignments. Conventionally, variant callers estimate the statistical likelihood of a genotype by modelling under the assumption of biallelic loci (Marth *et al.*, 1999). However, this can result in reduced variant calling accuracy within regions containing copy number variants or for polyploid organisms. *FreeBayes* overcomes this limitation by modelling under the assumption of multiallelic loci (Garrison and Marth, 2012). Candidate haplotypes are obtained using a dynamic window approach to identify proximal variants. The base and mapping quality is utilised to filter the candidate haplotypes. The length of indels are determined by increasing the window length for every overlapping read containing the haplotype information. The overlapping reads within the window are converged and used to estimate the posterior probability distribution for each genotype. The maximum

posterior probability and the genotype likelihood information are used to evaluate the probability that the locus is polymorphic, which is used to infer the genotype (Garrison and Marth, 2012).

The use of a haplotype-based variant detection method enables all variants to be evaluated in the same context and the use of phased genotyping can improve the detection of rare variants. The use of longer haplotypes can increase the signal to noise ratio for the genotype likelihood estimates, which improves variant calling (Garrison and Marth, 2012). *FreeBayes* has been shown to have the highest sensitivity in detecting variants in comparison to other variant callers such as *GATK HaplotypeCaller*. However, it has also been found to have low precision and to produce a greater number of false positive variant calls (Sandmann *et al.*, 2017).

Platypus

Platypus (Rimmer *et al.*, 2014) is another haplotype-based, assembly variant caller that integrates the methods employed by *GATK HaplotypeCaller* and *FreeBayes* to improve both the sensitivity and specificity of variant calling. *Platypus* identifies candidate haplotypes from a combination of read alignments against the reference sequence, local reassembly of the identified variants and known variants from SNP databases (see Figure 1.6). Local reassembly is carried out by constructing a colored de Bruijn graph, a variation of the de Bruijn graph which colours nodes according to the sample they belong to in order to preserve identity (Iqbal *et al.*, 2012), from the read and reference sequences. Candidate haplotypes are identified from unique paths within the graph and clustered together. Frequencies are estimated for all combinations of reads and candidate haplotypes $h_1, ..., h_a$ using an expectation-maximization algorithm (Dempster *et al.*, 1977) under the diploid genotype model as follows:

$$L(R|\{h_i, f_i\}_{i=1..a}) = \prod_s \sum_{i,j} f_i f_j \prod_{r \in R_s} \left(\frac{p(r|h_i)}{2} + \frac{p(r|h_j)}{2} \right) \tag{1.3}$$

where h_i is the haplotype, f_i is the haplotype frequency, a is the number of alleles, s refers to the sample, R denotes the read (R_s denotes the set of reads from sample s), i,j refers to the genotypes and $p(r|h_i)$ is the likelihood of the genotype. A likelihood matrix is computed for each read given the haplotype by aligning the reads against the haplotype sequence using the Forward algorithm. This algorithm computes the relevant joint probability by taking into account the probability of all the state paths, under the

hidden Markov Model. The frequencies are combined as a prior probability and the posterior support of the given haplotype is estimated from the likelihood of all possible haplotypes against the likelihood that the haplotype does not contain the variant as follows:

$$p(v|R) = \frac{p(v)L(R|\{h_i, f_i\}_{i=1..a})}{p(v)L(R|\{h_i, f_i\}_{i=1..a}) + (1 - p(v))L(R|\{h_i, \frac{f_i}{1-F_v}\}_{i \in I_v}} \tag{1.4}$$

where $p(v)$ is the prior probability for variant v, I_v is the set of haplotype indices in which the haplotype h_i does not contain the variant and F_v is the sum of haplotype frequencies for haplotypes that do not contain the variant. Variants are called when the posterior support exceeds the threshold Phred score of five. The posterior probability is calculated from the genotype likelihood by marginalising over other variants in the region and the maximum posterior probability is used to infer the genotype call (Rimmer *et al.*, 2014).

As *Platypus* utilises local reassembly for variant detection, it does not rely on the reference genome. This improves variant detection by reducing false positive variants that arise due to alignment errors within regions of greater sequence diversity. The process of identifying candidate variants and genotyping is carried out separately to reduce false negative variants. *Platypus* can also provide linkage information regarding local variants for further filtering and downstream phasing (Rimmer *et al.*, 2014).

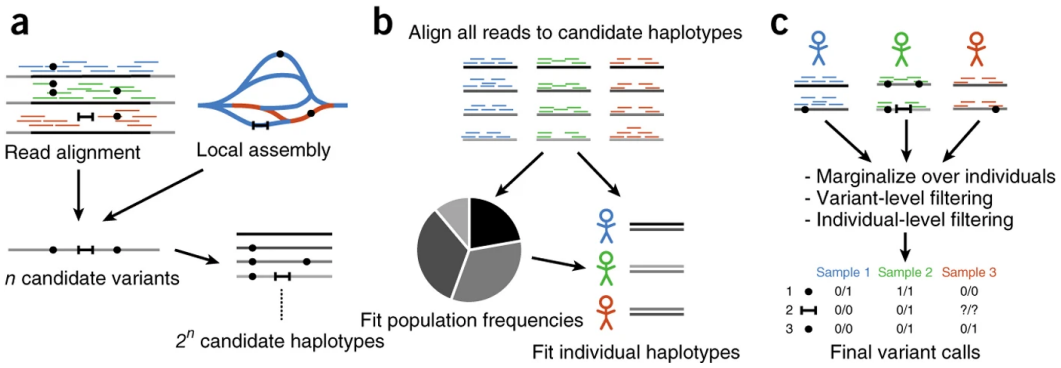


Figure 1.6: ***Platypus* Variant Calling Pipeline.** This figure shows the three stages of variant calling as follows: a) the identification of candidate haplotypes, b) alignment of reads against each haplotype is used to calculate population haplotype frequency and c) variants are marginalised across multiple samples and filtered on the posterior probability. This figure was published in Rimmer *et al.*, 2014.

Manta

Manta (Chen *et al.*, 2016), unlike other variant callers, has been optimised specifically for accurate discovery of large-scale structural variants (SV) and indels. *Manta* constructs a genome-wide breakend graph and utilises the edges to identify SV candidates. The graph edges do not represent a specific SV candidate but instead represent the possible junctions that connect two breakend regions, allowing for a compact representation of a genome. Each edge is used to identify SV candidates by searching over the mapped reads that span the connected nodes and discarding candidate variants that do not intersect the graph edge. The resulting candidate SVs are not associated with a precise genomic locus hence local read assembly is used to identify the precise region. Reads spanning the candidate SV region are assembled into contigs using a de Bruijn-like graph, aligned against the reference genome and candidate variants are refined on the alignment scores. The candidate SVs are scored using a diploid genotyping model to estimate the posterior probability of each candidate fragment and combined with the allelic likelihood estimate to infer the structural variant call. *Manta* allows for rapid detection of SVs by parallelising graph construction to reduce computational intensity. *Manta* was shown to have a high recall rate with a high fraction of true positive calls, allowing for accurate variant calling of structural variants (Chen *et al.*, 2016).

Limitations of Variant Calling

The use of posterior probability estimates to identify candidate variants provides a measure of statistical confidence. However, due to the presence of alignment and sequence artifacts, both pre- and post-filtering may be required to remove false positive variants. In the pre-filtering steps, trimming the raw sequence reads to remove adapter sequences and removing PCR bias by de-duplicating over-amplified reads can improve alignment (Nielsen *et al.*, 2011).

Alignment errors, most of which arise when aligning reads against genomic regions containing a greater degree of sequence diversity also reduce variant calling accuracy. Variant callers filter out reads with low mapping quality scores, which are assigned to reads that map to duplicated regions, such as copy number variants. This process can thus impede the discovery of true novel variants within the correctly aligned reads and result in false negative calls. Strand bias can occur due to the uneven distribution of

sequence reads between the forward and reverse strands leading to discrepancies between the inferred genotypes from both strands. This usually occurs in Illumina short-read sequencing data and can lead to an increase in false positive variants (Guo *et al.*, 2012).

Hard filtering can be applied post variant calling in order to reduce false positive calls by filtering out variants based on several constraints (see Section 4.1). This can be achieved using software such as *GATK Variant Quality Score Recalibration (VQSR)* (Van der Auwera *et al.*, 2013) and *VEF* (Zhang and Ochoa, 2019), both of which utilise machine learning algorithms. The cut-off values for filtering require careful consideration as stringent over-filtering can lead to a higher rate of false negatives due to the removal of true variants. As variant calling methods are heavily dependant on identifying variants from mapped sequence reads, it is essential to improve the accuracy of alignments against the reference in order to improve the accuracy of variant calling.

1.3 Genome Graph-based Reference Structures

The term "pan-genome" refers to the entire set of genes across a collection of multiple individuals within a species. It includes the core genome, genes that are conserved across all individuals and the accessory genome, genes that are only partially shared by specific individuals. The pan-genome for a species can be defined as either open, where each sequenced individual allows a number of new genes to be added, or closed based on the degree of diversity within a species and their capability to adapt and acquire new genes (Vernikos *et al.*, 2005). The use of pan-genomes has become commonplace in microbiology, particularly when studying bacterial genomes due to their large gene pool driving evolution via horizontal gene transfer, phage infection and uptake of genetic material from the environment (Medini *et al.*, 2005). The open pan-genomic nature of bacteria, due to their high genetic diversity, dictates that the use of a linear, single species reference is insufficient for studying genetic variability. Recently, due to the increasing feasibility of large-scale sequencing projects, the use of pan-genomes is also becoming routine when studying eukaryotes, including plants (*Arabidopsis thaliana* (Cao *et al.*, 2011); *Oryza sativa L.* (cultivated rice, Yao *et al.*, 2015); *Zea mays L.* (barley, Hansey *et al.*, 2012)), fungi (*zymoseptoria tritici* (Plissonneau *et al.*, 2018); *Saccharomyces cerevisiae* (Peter *et al.*, 2018; McCarthy and Fitzpatrick, 2019)) and humans (Li *et al.*, 2010). The Peter *et al.* study analysed the pan-genome across 1,011 *S. cerevisiae* iso-

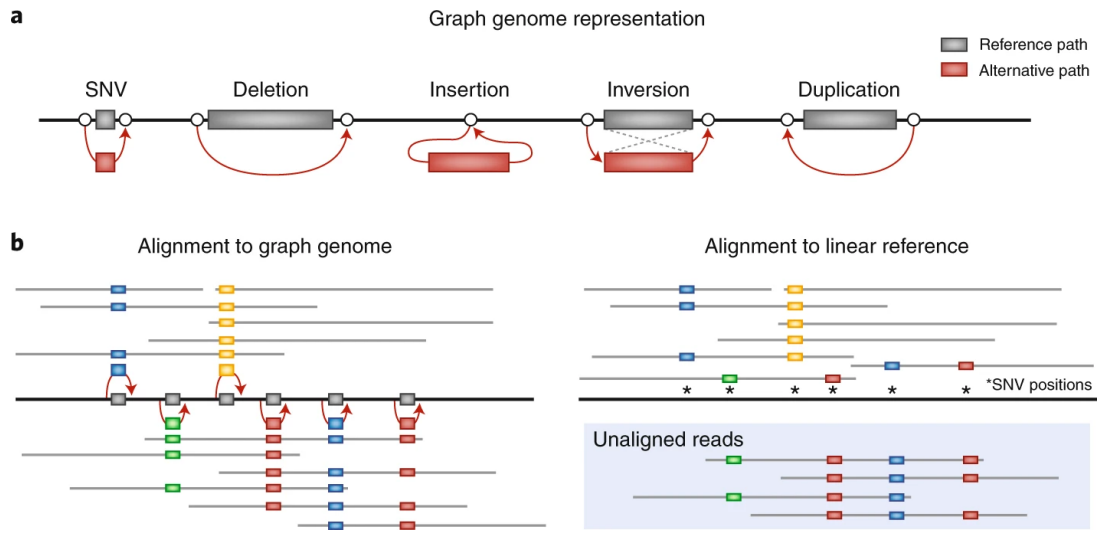


Figure 1.7: **Genome Graphs.** This figure is an example of a genome graph structure which shows how a) different variant types (shown by the red nodes) can be represented within the graph and b) how reads containing multiple SNPs may be able to align against the genome graph (left) but may not align against the linear reference (right). Fewer reads were able to align against the linear reference due to a threshold on the number of mismatches allowed in an alignment. This figure was published in Ameur, 2019.

lates and identified 4,940 core genes and 2,856 accessory genes. The accessory genes were found to have originated through horizontal gene transfer from other diverse yeast species and fungi, and tended to be clustered within the subtelomeric regions. This was reaffirmed in a more recent study (McCarthy and Fitzpatrick, 2019) which also identified a similar number of core and accessory genes across 100 *S. cerevisiae* strains. The latter study analysed the ancestral origins across the pan-genome, finding that the core genome was enriched for genes of prokaryotic origin whilst the accessory genome tended to be of eukaryotic origin with genes enriched for metabolism, pathogenesis and antimicrobial resistance.

A pan-genome can be represented graphically using genome graphs (see Figure 1.7), also referred to as variation graphs, enabling its use as the standard reference when encoding the genetic variability within a species, thereby offsetting the reference bias faced by the linear reference genome. Genome graphs have been used traditionally in sequence analysis for various purposes such as multiple sequence alignment and read assembly (Paten *et al.*, 2017). The reference genome itself forms the framework of the

graph onto which individual genomes can be incorporated as variants. A genome graph is defined as a graph that constitutes of a set of nodes, directed edges and embedded paths which describe the transformation of the graph into a genome sequence. Nodes, or vertices, within the genome graph represent smaller sets of genome sequences. An edge serves as the linkage between two nodes enabling the formation of a set of paths/walks, a series of connected nodes that describe the traversal through the graph in order to represent a larger genome sequence, through the graph. This allows for every single genome to be encoded via individual walks within the genome graph (Garrison *et al.*, 2018).

Genome graphs may be either cyclic or acyclic to reduce the computational complexity. A cyclic graph is one which allows for a node to be traversed more than once whereas each node is only encountered once within an acyclic graph. Genome graphs tend to be directed, where the edges encode the direction in which the nodes can be traversed, to allow for the order of nodes to imitate the sequence order. However, directed graphs are not able to portray strand information based on whether the sequence is read in its forward strand or as reverse complemented. Encoding this information is particularly important for displaying complex genome rearrangements such as reverse tandem duplications and inversions in the pan-genome. This issue, therefore, requires genome graphs to be bi-directed (Paten *et al.*, 2017). A bi-directed genome graph will enable an edge to enter on either side of a node based on the sequence orientation and leave the node through the opposite endpoint (Edmonds and Johnson, 2003). Insertions and substitutions within the graph are represented through the addition of new nodes containing the variant sequences and edges connecting the nodes to existing nodes within the graph. Deletions are represented through the addition of edges between existing nodes omitting the deleted sequence (Garrison *et al.*, 2018). Large genome rearrangements, such as translocations, can also be encoded within a genome graph as a deletion at one region and an insertion at another. However, a major limitation in this method of representation is that both the insertion and deletion will be treated as two separate events rather than as a single translocation (Goel *et al.*, 2019). Large structural variants and genome rearrangements have been shown to have a functional impact on complex disease phenotypes in humans, therefore, accurate representation of these complex variants is of utmost importance (Weischenfeldt *et al.*, 2013).

Due to increasing evidence that the use of graph-based reference structures provides incremental improvements in read mapping against the reference and allows for greater accuracy in variant calling, there has been an increase in the development of graph genome software since the *variation graph* toolkit (Garrison *et al.*, 2018) was first released. Each of the graph genome software implement novel algorithms for graph construction, augmenting variants, indexing, querying for read alignments and variant calling. Many of the ideas that led to the conceptualization of these software originated from a pilot study conducted by the Global Alliance for Genomics and Health (GA4GH) which tasked various global teams to develop novel algorithms for genome graph construction. This led to the creation of eight different pipelines that employed differing methodologies such as de Brujin graphs, multiple genome alignments and k-mer based HMM approaches. This was also the first comprehensive study that tested the performance of read mapping and variant calling using graph genomes, showing that they perform better than the linear reference genome (Novak *et al.*, 2017). The various methods for graph genome construction and genotyping utilised by each software are detailed below.

1.3.1 *Variation Graph* Toolkit

The *variation graph* (*vg*) toolkit (Garrison *et al.*, 2018) arose from the initial study by GA4GH and was the first graph genome implementation to be released. The *vg* toolkit was built with the purpose of creating, manipulating and utilizing variation graphs as reference structures. Variation graphs are constructed using a FASTA file containing a linear reference genome and a VCF (Danecek *et al.*, 2011) file containing variants from the population. However, *vg* does allow for variation graphs to be constructed using methods similar to that of *de novo* assemblers in the absence of a reference genome. The order of nodes within the variation graph represents the forward strand which is used to form walks through the graph. Valid walks through the graph are made by leaving nodes from the opposite side from which they were entered. The variation graphs constructed by *vg* are bi-directed, thus nodes can also be entered from the opposite side in order to read the sequence as reverse-complemented. Divergent nodes in the variation graph indicate the presence of non-reference alleles at that locus. Variation graphs also have the ability to display large-scale structural variants such as inversions, duplications and copy number variation, especially useful within highly repetitive regions (Novak *et al.*,

2017).

The variation graph is indexed using a Succinct Data Structure Library (SDSL) so that it can be stored in a memory efficient manner and the GCSA2 index uses rank/select dictionaries to allow for querying of the graph (Garrison *et al.*, 2018). Rank/select dictionaries are data structures that store the variation graph in a BWT suffix tree along with the rank (the number of times that element occurred in the array) and select (the position of the element) operations for each element after sorting (Raman *et al.*, 2007). Alignment of reads against the graph uses a seed and extend alignment model. The GCSA2 index suffix tree is traversed to find seed matches, and seeds in close proximity are clustered before carrying out local dynamic alignment of the reads against the clusters. Super-maximal exact matches (SMEMs) are identified and clustered together. Optimal alignments are identified using a Markov model to calculate a maximum likelihood path. In reads where the MEMs do not extend across the full read length, the Markov model is again applied to identify the highest-scoring chain of MEMs. The Markov model rewards long SMEMs, and SMEMs with shorter gaps in between, for the selection of candidate alignments. Once the maximum likelihood path is obtained, a directed acyclic subgraph is extracted from the reference for each local region and reads are aligned using a banded graph striped Smith-Waterman algorithm. Finally, alignment scores are calculated for each mapping, taking into account the Phred quality score and the mapping quality score (Garrison *et al.*, 2018).

Variant calling can be carried out on aligned reads to identify any variation in sequence from the graph. Before variant calling, reads with secondary/ambiguous mappings are filtered out. The graph is then extended into an augmented graph with novel alleles from the alignments. The amount of read support at each position, also referred to as pileup, is calculated. Variant calls are made by thresholding reads and filtering out alleles with little to no read support. The variation graph is unique in that the called variants can be incorporated back into the graph through editing operations in *vg* without the need to construct a new variation graph (Garrison *et al.*, 2018).

The *vg* toolkit has been shown to align both human and yeast genomes with greater sensitivity and a lower false discovery rate in comparison to the linear reference genome (Garrison *et al.*, 2018). An advantage of the *vg* toolkit is that the variation graphs can

be visualised, albeit only for short regions of the genome, which is greatly beneficial in the understanding of how variant alleles are encoded within the graph. A major limitation of the software is the huge computational memory requirements necessary for generating the graph index. Index generation for the human genome was found to require 1.5 Tb of memory, which may not be available to many users. In addition, the GCSA2 index generation requires for the complexity of the graph to be reduced by removing nodes with a degree of bifurcation higher than a specified threshold within high complexity regions. As these regions are likely to be highly polymorphic, reducing their complexity may impact on the accuracy of read alignments within these regions.

1.3.2 *Seven Bridges Graph Genome* Toolkit

The *Seven Bridges Graph Genome* toolkit (Rakocevic *et al.*, 2019), also known as the *Graph Genome Pipeline (GGP)*, was also developed from the GA4GH study. It aims to overcome the limitations faced by other graph genome software, which due to the expensive time and memory requirements, limits graph construction to either very small genomes or smaller portions of larger genomes. The Graph Genome toolkit provides a computationally efficient method for whole-genome graph construction, read mapping and variant calling.

A graph genome is constructed by storing sequences within vertices connected together via edges (see Figure 1.8). An adjacency list data structure is used to store all the connected edges for each vertex. An adjacency list is an unordered array of linked lists where the length of the array directly corresponds to the number of vertices within the graph. Each vertex contains a linked list of references to other vertices with a shared edge (Jain, 2015). The sequences are compressed into blocks based on their size and stored in a buffer separately. The graph structure is serialized by storing the start and end loci and a reference to the sequence for each edge. Storing the edge information optimises the speed of graph manipulation as adding new variants involves the insertion of new edges (Rakocevic *et al.*, 2019).

A hash table is used to index the graph genome to allow for efficient querying of sequence reads during alignment. The graph is traversed sequentially to generate k-mers for all possible paths in the graph. Hash values are computed based on the

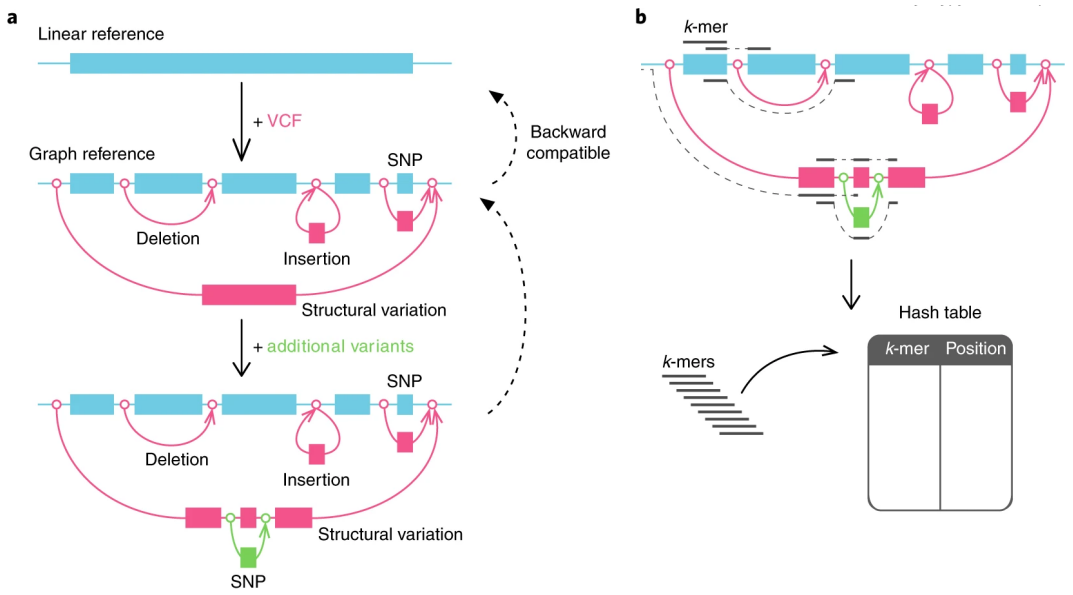


Figure 1.8: **Seven Bridges Graph Genome Toolkit.** This figure demonstrates a) how the variation graph is constructed to incorporate variants and can be augmented with additional variants and b) how the graph is indexed by generating k-mers of all possible paths in the graph and storing the locations within a hash table. This figure was published in Rakocevic *et al.*, 2019.

sequence for each k-mer to obtain the index key for the hash table and its position is stored as the value. Backward compatibility to the reference position is maintained in indexing by asserting the position for any variant edge as having taken the reference path through the graph leading up to the branching edge. Index positions containing many values indicating very common sequences are initially removed to seed the more unique sequences first. Sequence reads are searched against the graph by generating hash indexes for k-mers within the reads and querying the hash table to obtain a list of all k-mer loci. Candidate match regions are identified by using a sliding windows approach to locate for loci clusters corresponding to the list. Clusters are scored by matching the positions of k-mers in the sequence read against the positions in the graph and any cluster exceeding the threshold score is identified as a seed for alignment. First, gapless local alignment of candidate seeds is carried out using the graph-aware bit-parallel approximate (BPA) algorithm, a fast approximate string matching algorithm. The BPA algorithm parallelises computation by encoding the differences between two strings using bit-vector operations (Hyyro, 2005) based on the Levenshtein distance, the minimum number of single-character edit distances required to transform one string into another (Levenshtein, 1966). If k-mers at the ends of a sequence read are not present within

the graph index, suggesting the presence of insertions or structural variants, the graph region is extended on either side of the read. If the BPA algorithm is unable to align a read with fewer than four mismatches, a custom SIMD-optimised Smith-Waterman algorithm that permits gapped alignment is used to re-align the read (Rakocevic *et al.*, 2019).

Variants are called by utilising the CIGAR strings from the BAM file to identify regions with candidate variants using 300 bp windows. Overlapping reads are compressed into a de Bruijn-like graph and candidate haplotypes are derived by finding variants with the largest read support and scoring them using pair Hidden Markov Models. The graph reference genome is used to give variants within the graph a higher prior probability in order to overcome reference bias (Rakocevic *et al.*, 2019).

The *Graph Genome Pipeline* was shown to improve read alignment and the accuracy of variant calling, especially across large-scale structural variants and their break-point regions when compared against the linear reference genome for human genomes (Rakocevic *et al.*, 2019). However, the toolkit is inaccessible to many users due to its hardware requirements, as the Docker container files needed to run the software require a Linux operating system with AVX2 instruction set processing capabilities. Another limitation is that the graph is constructed in memory, therefore, cannot be accessed or visualised. As a result, the accurate incorporation of variants within the graph cannot be easily confirmed.

1.3.3 *BayesTyper*

The *BayesTyper* (Sibbesen *et al.*, 2018) software provides another variant-aware graph genome method that focuses on improving both the sensitivity and accuracy of genotyping. The significant loss of information due to the lack of read alignments against variant-dense regions makes genotyping unreliable, especially for large-scale structural variants. *BayesTyper*, therefore, takes a two-stage approach to unbiased genotyping which requires the initial alignment of sequence reads against a linear reference genome (see Figure 1.9). A combination of variant callers (*GATK HaplotypeCaller* for detecting SNPs (McKenna *et al.*, 2010), *Platypus* for medium-sized variants (Rimmer *et al.*, 2014) and *Manta* for large-scale indels (Chen *et al.*, 2016)) are used to generate candi-

date variants from the alignment which are subsequently combined with known variants to create a database of variant candidates. The prior variant database and the linear reference genome form a variation graph against which the sequence reads are realigned to allow for accurate genotyping (Sibbesen *et al.*, 2018).

K-mer profiling is carried out across the sequence reads using the KMC3 program (Kokot *et al.*, 2017) to count *k*-mers of length k=55, thereby allowing *k*-mers from an individual to be compared against *k*-mers from candidate variants in the database. Variants whose breakpoints are less than *k*-1 nucleotides apart, and therefore share a *k*-mer, are clustered together. A variant graph is constructed for each variant cluster based on the candidate variant database and the linear reference genome, where the variant sequences are represented as divergent nodes at each locus. Each variation graph is used to generate candidate haplotypes, based on *k*-mer occurrences, using the n-best heuristic algorithm. The n-best search algorithm uses similar principles to the Viterbi algorithm to identify the n most likely paths (Chow and Shwartz, 1989) instead of a single maximum likelihood path. Prior to searching, the ntHash algorithm, developed specifically to hash consecutive *k*-mers within sequence reads (Mohamadi *et al.*, 2016), is used to construct bloom filters from the *k*-mer profiles. Bloom filters are a probabilistic data structure that informs whether a given *k*-mer is definitely absent or may be present within a sequence. The bloom filters are used to rank the paths within the graph identified with the n-best algorithm using a two-tier scoring system. First, the paths are ranked by the number of *k*-mers present within the path that are also seen in the bloom filter then secondly ranked by the number of nodes observed within a path that is not present in a higher ranking path, with the rankings used to select the 16 best paths. The graph is then traversed to rank the paths at each node before the best paths that cover the greatest number of nodes are merged to generate a set of candidate haplotypes for each cluster and a read *k*-mer count table of each haplotype *k*-mer for each individual (Sibbesen *et al.*, 2018).

As BayesTyper was created mainly for genotyping human genomes, by default the software aims to identify diplotypes from the read *k*-mer counts. However, the ploidy level can be specified for haploid organisms. The genotype is inferred by modelling an individual's diplotype as being drawn from a population of haplotypes and there-

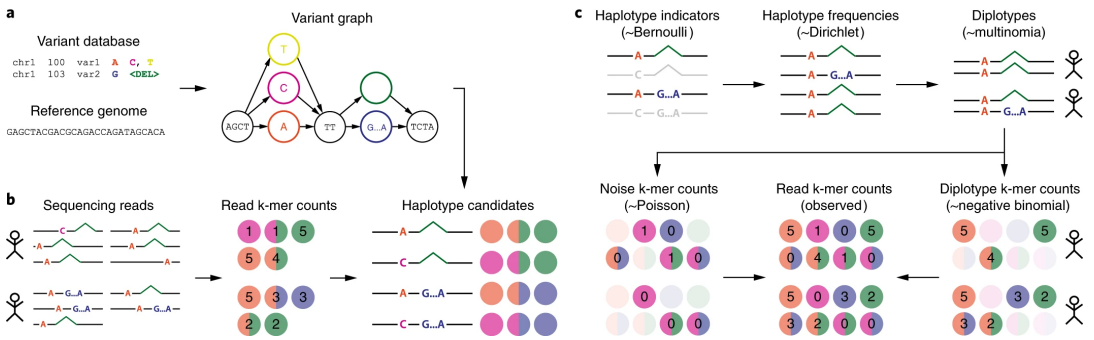


Figure 1.9: ***BayesTyper.*** This figure illustrates the methods utilised by *BayesTyper* for inferring genotypes. a) The variation graph is constructed using known variants and a prior variant database. b) k-mer profiling is carried out on the sequence reads and candidate haplotypes are identified by scoring based on the k-mer counts. c) The genotype is inferred under the assumption of a diploid genome by taking into account the k-mer count for that individual and noise k-mer counts. This figure was published in Sibbesen *et al.*, 2018.

fore expects only a subset of the haplotypes to be present within the population. The expected sparsity in the haplotype frequency is modelled by applying a sparse prior to the Dirichlet distribution. The population frequency is used to draw diplotypes for each individual by modelling under the negative binomial distribution as the observed k-mer counts being generated from a combination of the sampled diplotype, non-variant regions between the variant clusters and noise due to sequencing errors. The final genotype is inferred by calculating the posterior probability distribution for the diplotypes based on the k-mer count using Gibbs sampling (Sibbesen *et al.*, 2018).

The major limitations of using the *BayesTyper* pipeline when working with yeast genomes is that it does not support genotyping for individuals with a ploidy level greater than two. Consequently, it can only be used to genotype haploid or diploid strains, limiting its functionality. In addition, *BayesTyper* cannot be used for variant discovery, making it redundant when aiming to identify novel variants or genotype species with a lack of prior variant information.

1.3.4 *Graphtyper*

Graphtyper is another software that proposes a different method of graph construction to allow for improved sensitivity in both genotyping and variant discovery. As with *BayesTyper*, *Graphtyper* requires the initial alignment of sequence reads against a linear reference genome and due to their large size, only works with small portions of the reference genome rather than the whole genome. It was created mainly for the purpose of genotyping human genomes. However, in order to reduce mapping bias and increase the accuracy of alignments around indels, sequence reads are realigned against a variation graph. This enables true variants to be distinguished from sequencing errors within reads, reducing the discovery of false positive variants (Eggertsson *et al.*, 2017).

The pan-genome variation graph is constructed using the linear reference genome and known variants for the specified region (see Figure 1.10). Any variants with overlapping reference alleles are merged in the initial stage before constructing the variant nodes and storing the starting positions of the variant allele. The reference nodes are constructed between adjacent variant nodes and the starting positions are again stored. The nodes are connected to form a directed acyclic graph containing the sequence paths for each genome. The graph is indexed by generating k-mers of length k=5 for all paths through the graph. The starting and ending positions in the reference genome and the id of overlapping variant nodes are also stored (Eggertsson *et al.*, 2017).

Initial alignment against the linear reference with *BWA* is utilised to report the sequence reads that aligned to that specific region. These reads are realigned against the variation graph by generating consecutive k-mers across the sequence reads and checking against the index table to ensure the k-mers are present. Seeds are identified based on k-mer matches in the index lookup and extended across adjacent k-mers that also matched. The longest seeds are extended by using a search algorithm to determine the path with the fewest mismatches. If a seed cannot be extended with 12 or fewer mismatches, the process is repeated by re-extracting k-mers. A single mismatch is allowed in each k-mer until the longest seed is found and realigned against the variation graph (Eggertsson *et al.*, 2017).

Variants that are located within 5 bp distance of each other are clustered together and each individual cluster is genotyped independently. For each candidate haplotype,

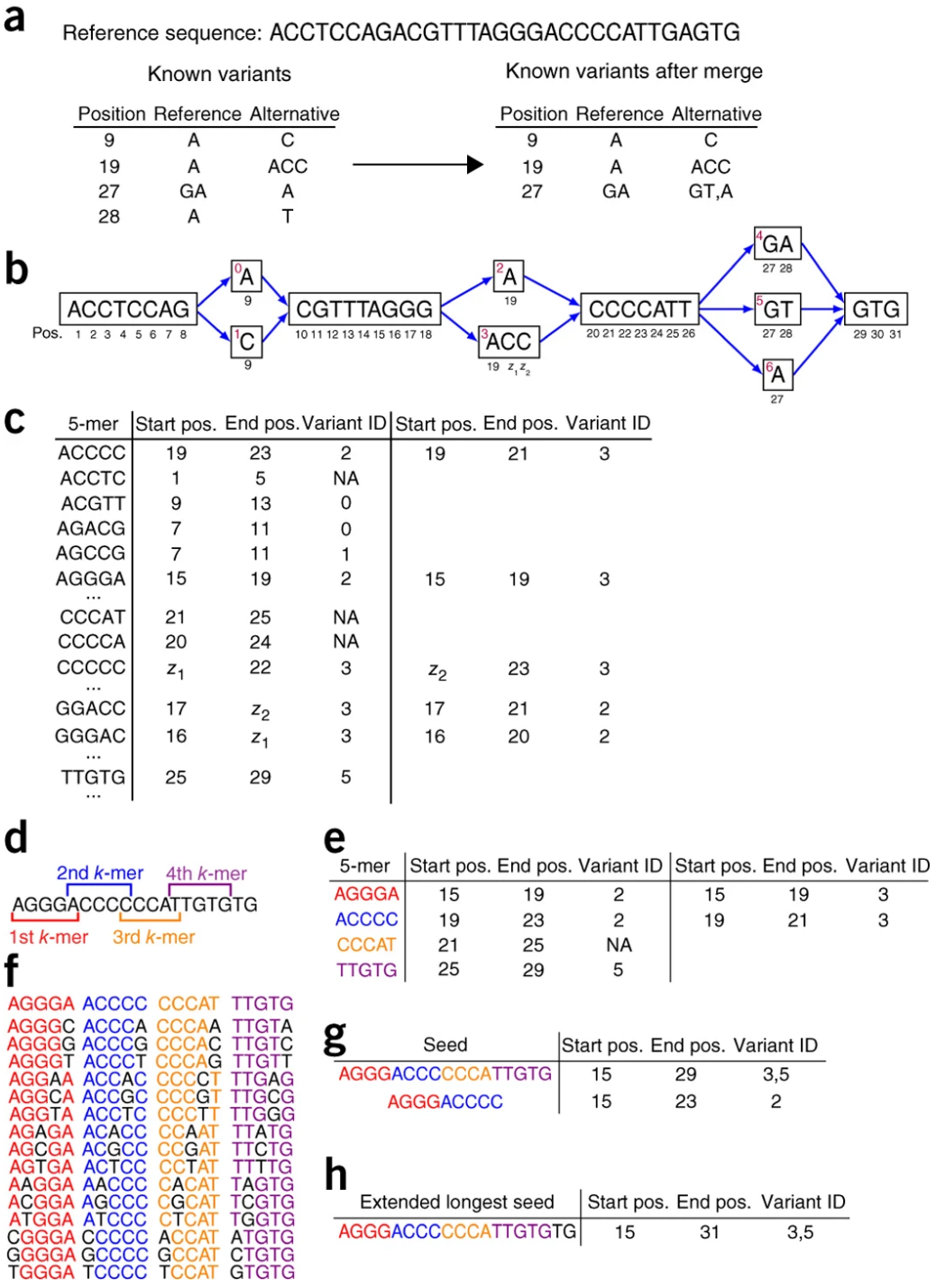


Figure 1.10: ***Graphtyper.*** This figure is an overview of the Graphtyper pipeline. a) Overlapping variants are merged together. b) A variation graph is constructed. c) The graph is indexed by generating k-mers of length 5 and storing the starting and ending positions. d) K-mers are generated from the sequence reads. e) The k-mers are queried against the index table. f) k-mers with a single base substitution are extracted. g) Seeds that match against the index table are identified. h) The longest seeds are extended to carry out read alignment. This figure was published in Eggertsson *et al.*, 2017.

the relative likelihood of observing a sequence read within the population given the haplotype is estimated under the assumption that reads from each sample are independent of one other. The relative likelihood is estimated based on the base quality, the mapping quality of the sequence read and whether the reads are soft-clipped. The genotype is then inferred based on the haplotype with the highest likelihood. Any unobserved haplotypes are also removed from the graph to reduce graph complexity. The reference sequence is extended by 50bp on either side and local realignment of the read is carried out using a banded semi-global version of Gotoh's algorithm. Gotoh's algorithm is a dynamic variation of the Smith-Waterman algorithm that optimises affine gap scoring to produce a single optimal alignment (Gotoh, 1982). Any observed differences in the local alignment are determined to be variants and novel variants are called if at least five observations with an alternate allele frequency of 0.2 are made (Eggertsson, 2017).

Sequence reads are selected for realignment against the variation graph based on the regions within the linear reference to which the reads aligned. This means that *Graphtyper* shares certain limitations faced by the linear alignments. There is still a loss of information from reads that are completely unable to align against the variant-dense regions of the linear reference, as the genomic region for these reads will not be reported. Unlike *BayesTyper* and the *Graph Genome Pipeline*, the variation graph is not created in memory but similarly the graph cannot be accessed for visualisation. In addition, it requires several repetitions of the pipeline to be run in order to construct a variation graph across the whole genome.

1.3.5 Advantages of Variation Graphs as Reference Structures

There are several advantages to using variation graphs as reference structures as they are able to overcome many of the limitations of a linear reference genome. Variation graphs could be used for genomic analyses in species where there is yet to be a reference genome as graphs can be constructed without a reference. The presence of prior variants within the variation graph can mitigate reference allele bias in mapping. Variation graphs can also increase the proportion of reads that are able to map against the reference, especially within the hypervariable regions where there is a large number of non-reference alleles. The increased ability to map against polymorphic regions is essential when studying yeasts as they display a great level of intragenic variation. Vari-

ation graphs have also been found to produce less ambiguous mapping as the inclusion of variants enabled reads to distinguish between true mapping location and secondary, paralogous locations (Novak *et al.*, 2017).

1.4 Introduction to Yeast

Yeasts are primarily single-celled micro-organisms, typically ranging from 4-6 μm in size, that belong to the kingdom Fungi under the domain Eukaryota. Yeasts display a high level of taxonomic diversity and can be placed within two different phyla: Ascomycota and Basidiomycota, which together form the subkingdom Dikarya. Ascomycota contains the greatest number of fungi including the true yeasts, also known as the budding yeasts (see Figure 1.11). Most species of yeast undergo asexual reproduction by mitosis to form new yeast cells through either budding e.g. *Saccharomyces cerevisiae* or fission e.g. *Schizosaccharomyces pombe*. Yeasts are predominantly unicellular but favourable conditions can cause rapid budding to form a string of connected buds known as the pseudohyphae (Kurtzman and Fell, 2006). During stressful conditions under nutrient depletion, haploid cells are unable to survive therefore yeasts mate and reproduce sexually through meiosis. The resulting diploid cells sporulate to form ascopores which contain from four to eight haploid cells encapsulated within the sac-like ascus (Neiman, 2005).

Over 1,500 species of yeasts have been identified to date from a diverse range of natural habitats such as salt water, plant leaves and soil. Yeasts are also found on skin surfaces or inhabiting the genitourinary and gastrointestinal tracts of animals where they exist commensally as a part of the gut microbiota. Even though most species of yeast are non-pathogenic, certain *Candida* species are opportunistic pathogens which cause fungal infections when the host becomes immunocompromised (McManus and Coleman, 2014). *Candida albicans* is the most clinically relevant of the yeast pathogens as it has the highest prevalence in cases of both systemic (Stappers and Brown, 2017) and oral (Thompson *et al.*, 2010) candidiasis.

Yeasts were one of the earliest organisms to be domesticated due to their ability to ferment sugars. Yeasts can be either obligate aerobes that require oxygen or facultative anaerobes that are able to switch between the respiratory pathways. They utilize car-

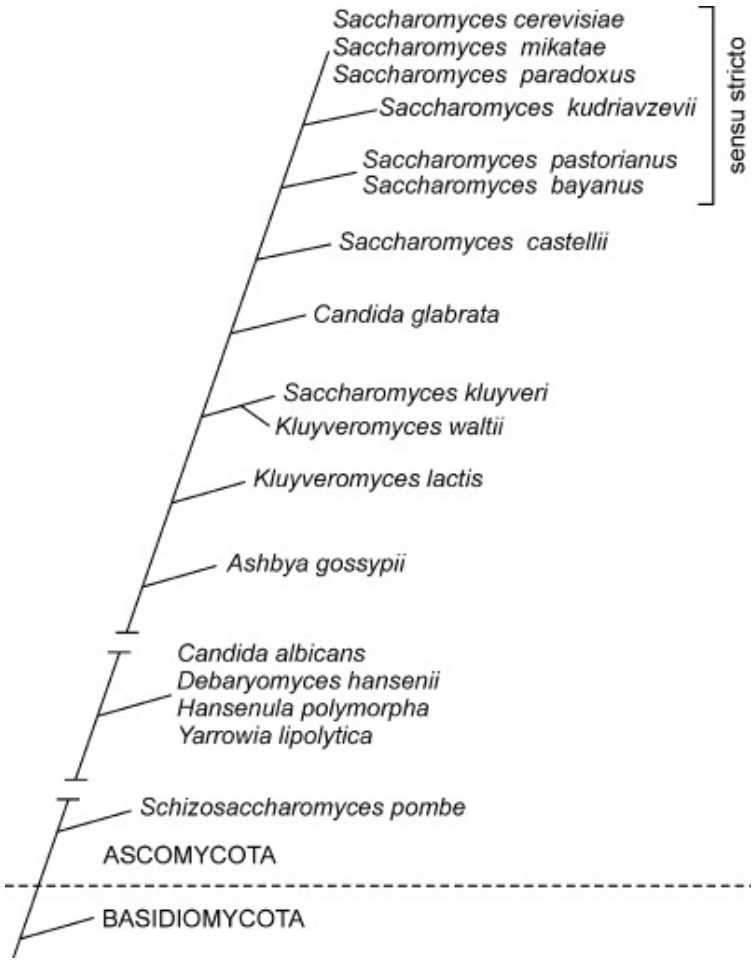


Figure 1.11: **Highly Profiled Yeast Species.** This figure shows the relative phylogenetic relationship across highly profiled yeast species since the divergence of Ascomycota from Basidiomycota approximately 550 million years ago. This figure was published in Piškur and Langkjaer, 2004.

bohydrates as an energy source during fermentation to convert into carbon dioxide and alcohol in the absence of oxygen. The process of fermentation has been used for centuries in baking where yeast is a leavening agent and in brewing for production of beer and wine. *Saccharomyces cerevisiae* is the most commonly used species of yeast due to its ability to adapt to different environments and metabolise a range of carbohydrates.

Yeasts are highly important organisms that have had a significant impact on human history. The domestication of yeasts and the ability to control their environment has allowed for them to be exploited in many fields of industrial biotechnology. Hence, understanding their physiological properties is essential for bioindustrial development.

For example, *Yarrowia lipolytica*, a dimorphic ascomycetous yeast species, has the ability to degrade hydrocarbons and aromatic compounds (Zinjarde *et al.*, 2014) giving it environmental applications in waste treatment and crude oil bioremediation (Bankar *et al.*, 2009). Another environmental impact of yeast fermentation is the production of ethanol from feedstock in the biofuel industry to reduce crude oil consumption (Azhar *et al.*, 2017). Yeasts also play a significant role in biomedical research. *Saccharomyces cerevisiae* is used as the model organism on which to study cellular mechanisms in eukaryotes. The conservation of genes between humans and yeast has enabled discovery of novel proteins and yeast has been used as a vector for non-native proteins. *S. cerevisiae* is also the most commonly used species in the various bioindustries, therefore, its functional characterization can improve genomic research and optimize bioindustrial production of a range of useful chemicals.

The *S. cerevisiae* genome is small and highly compact, with genes accounting for approximately 72% of the whole genome. It is approximately 12.1 Mb in length and organized into 16 chromosomes consisting of 6,275 genes and an extranuclear mitochondrial genome. An estimated 23% of the protein-encoding genes in *S. cerevisiae* are found to have human homologs (Saraswathy and Ramalingam, 2011). The *S. cerevisiae* genome also contains extensively repeated regions which are difficult to sequence. These highly repetitive sequences such as the ribosomal DNA, retrotransposons and subtelomeric regions, are functionally important and therefore can be used to study evolution and variation across the yeast species. The *S. cerevisiae* haploid laboratory strain, S288c, was developed through numerous deliberate hybridisation events, carried out by both Carl Lindegren and Robert Mortimer, to be utilised for the isolation of biochemical mutants. It was the first *S. cerevisiae* strain to be sequenced and thus, it is used as the reference genome to guide the alignment of sequence reads (Engel *et al.*, 2014).

The National Collection of Yeast Cultures (NCYC) contains a highly extensive collection of yeasts that have been isolated from a range of geographical locations across the globe. The NCYC was established in 1938 and has acquired over 4,000 strains of yeast, with around 25% of the collection comprising of *S. cerevisiae* strains, over the eight subsequent decades. The collection contains largely non-pathogenic species of yeasts involved in a whole range of functions such as brewing, baking, genetically-defined strains for biomedical research and strains that can be selected based on specific

character traits. The NCYC genome sequencing project has enabled the genomes of approximately 1,000 strains, including around 400 *S. cerevisiae* strains, to be sequenced. There is a wealth of information contained within the NCYC strains that could be further explored through functional analysis. Several *S. cerevisiae* strains within the collection have already been highly profiled yet characterisation of the genetic variants amongst many of the strains is still required to improve our understanding of the species.

1.5 Aims and Objectives

The fundamental aim of this project is to establish computational methods that could be utilised to optimise the accuracy of variant prediction in yeast genomes. Continuous efforts have been made to allow for increased precision in the identification of genetic variants from next-generation sequencing data by improving upon the accuracy of existing methods. However, despite these efforts, the alignment of sequence reads against the linear reference genome has been recognised to impair the accuracy of all subsequent analyses. As such, this project aims to compare whether the use of a pan-genome variation graph reference structure can improve read mapping. The initial objective is to understand how variation graphs are constructed and to compare the quantity and quality of alignments against the linear reference genome using both NCYC and third-party *S. cerevisiae* strains. A few different implementations of variation graph software have been released, however, there is yet to be a study that extensively compares the performance of these software. Thus, another key aim of this project is to carry out a comprehensive study of the performance of the mapping and variant calling algorithms amongst four graph genome software against the linear pipeline. The final aim is to explore read filtering on alignments as a novel technique to remove false positive variants prior to variant calling.

Chapter 2 focuses on the construction of variation graphs using the *vg* toolkit to understand how variant alleles are embedded within the graph. It also examines how the variation graph can be used to indicate the degree of intragenic variation within a population. The quantity and quality of alignments against the variation graph are assessed in comparison to the linear reference genome. A simulation study is carried out to understand how various factors impact read mapping ability in *vg*. Chapter 3 looks at developing the FAT-CIGAR software to obtain exact alignment information for

graph-based and linear read mappers. It examines the mapping quality in alignments from *vg*, the Graph Genome toolkit and *BWA* based on sequence identity scores generated from the FAT-CIGAR string. The variant calling accuracy across the four graph genome software is assessed by comparing the precision, recall and F1 scores across 1,000 simulated datasets. Chapter 4 highlights the FAT-CIGAR toolkit as a novel method of variant filtration that removes false positives by anchoring read ends against the reference sequence by a specified number of bases. It also introduces the *sim_genomes* program for genome simulation with greater control over variant induction. It compares the performance of the filtered reads across simulated genomes with differing ratios of SNPs and indels to identify the optimal read filtering. It also discusses the potential to remove false positive variants from real datasets, examining three NCYC strains whose genomes were sequenced in duplicate. Chapter 5 provides an overview of the findings in this project and discusses potential future work that can further optimise the accuracy of variant prediction.

Chapter 2

Transitioning Towards the Use of Variation Graphs as Reference Structures

2.1 Introduction

It has been proposed that read mapping to a graph-based reference is more accurate than to a linear reference (Novak *et al.*, 2017, Garrison *et al.*, 2017). The aim of this chapter is therefore to understand how variation graphs are constructed by the graphical *vg* software using haploid *S. cerevisiae* strains and to visualise how variant alleles within a strain population are represented within the resulting variation graph. It will highlight how the complexity of the variation graph can provide an indication of the level of intragenic variation within each chromosome by looking at the node to sequence length ratio. The performance of the *vg* mapping algorithm will be evaluated against the conventional mapper, *BWA*, both in terms of the quantity of mapped reads and the quality of alignments which will be determined by the alignment scores. The differences in the scoring systems employed by both algorithms will be explored identifying the need for a suitable metric to determine alignment quality from different software. The *vg* mapping algorithm will be further studied by looking at factors that impact read mapping by simulating reference genomes of varying lengths and mapping sequence reads with different types and size of mutations and assessing the quantity of mapped reads.

2.2 Methods

2.2.1 Constructing Variation Graphs

The following eight *Saccharomyces cerevisiae* strains were selected from the National Collection of Yeast Cultures (NCYC) strain collection in order to gain more in-depth understanding about constructing variation graphs using the *vg* v1.5 software (Garrison *et al.*, 2018): NCYC78, NCYC88, NCYC93, NCYC97, NCYC214, NCYC221, NCYC222 and NCYC230. All of the strains were chosen due to their high sequence read quality and were haploids, thereby ensuring that differences in ploidy levels did not affect the construction of the variation graph. The strains were sequenced using an Illumina HiSeq sequencer as part of a large-scale sequencing project carried out by the NCYC. The raw sequencing reads of the NCYC datasets were pre-processed by Dr Jo Dicks. First, *BBTools* v38.47 (Bushnell, 2014) with the *clumpify* option was used to remove PCR duplicates. Second, regions of low quality and any remaining adapter sequences were removed from the de-duplicated reads using *Trimmomatic* v0.32 (Bolger *et al.*, 2014; default parameters and adapter sequence files relevant to the sequencing library used for each strain). Third, FreeBayes v1.2.0-4-gd15209e (Garrison and Marth, 2012) with default parameters was used to predict sequence variants, with the resulting VCF files filtered for binary, high-quality (Q>30) SNPs in *R* v3.1.1 (R Core Team, 2017) using custom scripts. As *vg* requires a single VCF file for graph construction, the eight VCF files were merged into a single multi-sample VCF file using the *GATK* v3.8-0 *CombineVariants* tool (McKenna *et al.*, 2010). The multi-sample VCF file was compressed into BGZF format with the *bgzip* program and indexed using the *Tabix* program (Li Heng, 2011).

The variation graph was constructed using the *vg construct* function with the S288c *S. cerevisiae* reference FASTA file and the multi-sample VCF file as input (see Figure 2.1 for a summary of the methods utilised in *vg*). The *-R* parameter was specified to limit graph construction to each specific chromosome due to the large memory requirements for whole-genome graph construction in *vg*, producing 17 variation graphs for each of the 16 chromosomes and the mitochondrial genome. The paths in each graph were checked to ensure they contained the correct chromosome using *vg paths* and validated using the *vg validate* function to ensure the graphs did not contain errors such as unconnected nodes and orphaned edges. The 17 chromosome graphs were then passed through *vg ids* to generate a joint id space across the nodes, ensuring they were treated as if they

were one single graph. The XG and GCSA2 (Generalized Compressed Suffix Array) indexes, which store the graph information succinctly, were generated with *vg index*. The XG index contains the structure of the graph but does not conserve the actual sequence. The GCSA2 index contains a suffix array that allows querying of specific sequences in the graph which is necessary for sequence read mapping. Before GCSA2 index generation, the computational complexities of the graphs were reduced by pruning them to remove any edges that induced more than 4 bifurcations within a sequence of length 16 bp. This was achieved by using the *vg mod* function with the parameters *-pl 16 -e 3*. The resulting subgraphs and orphaned edges were stripped out using *vg mod* with the parameters *-S -l 32*. These lowered complexity graphs were overlayed onto a graph containing only the S288c reference sequence using *vg mod* with the parameters *-N -t 32*, to ensure there were no missing id spaces. De Bruijn graphs were constructed from the reduced complexity graphs with k-mers of length k=16 using *vg kmers* and provided as input for GCSA2 index generation.

A sequence dataset derived from the tetraploid *S. cerevisiae* ale strain, NCYC1006, was chosen to be mapped against the haploid variation graph to evaluate how well the *vg* mapping algorithm aligns reads when there are differences in ploidy level between the strain and the graph. Read mapping against the variation graph was carried out using *vg map* with the NCYC1006 interleaved FASTQ file (containing both read pairs in a single file), the XG and GCSA2 index as input. The subsequent read alignment information from the mapping procedure was written out to a GAM (Graph Alignment/Map) file, a highly-detailed *vg* equivalent of the BAM file produced by *BWA* (Li and Durbin, 2009). As the GAM file format is binary, read information and their corresponding alignment scores were parsed out into human-readable CSV file format using the *jq* v1.4 (Dolan, 2014) software. The CSV file was loaded into the *R* v3.4.2 software and a histogram was utilised to study the distribution of alignment scores and to analyse how well the reads mapped.

2.2.2 Comparison of Mapping Efficacy Between Linear and Graph-based Approaches

Here, we aimed to replicate previous studies comparing the efficacy of read mapping when using variation graphs, both when variants are included and excluded (the latter

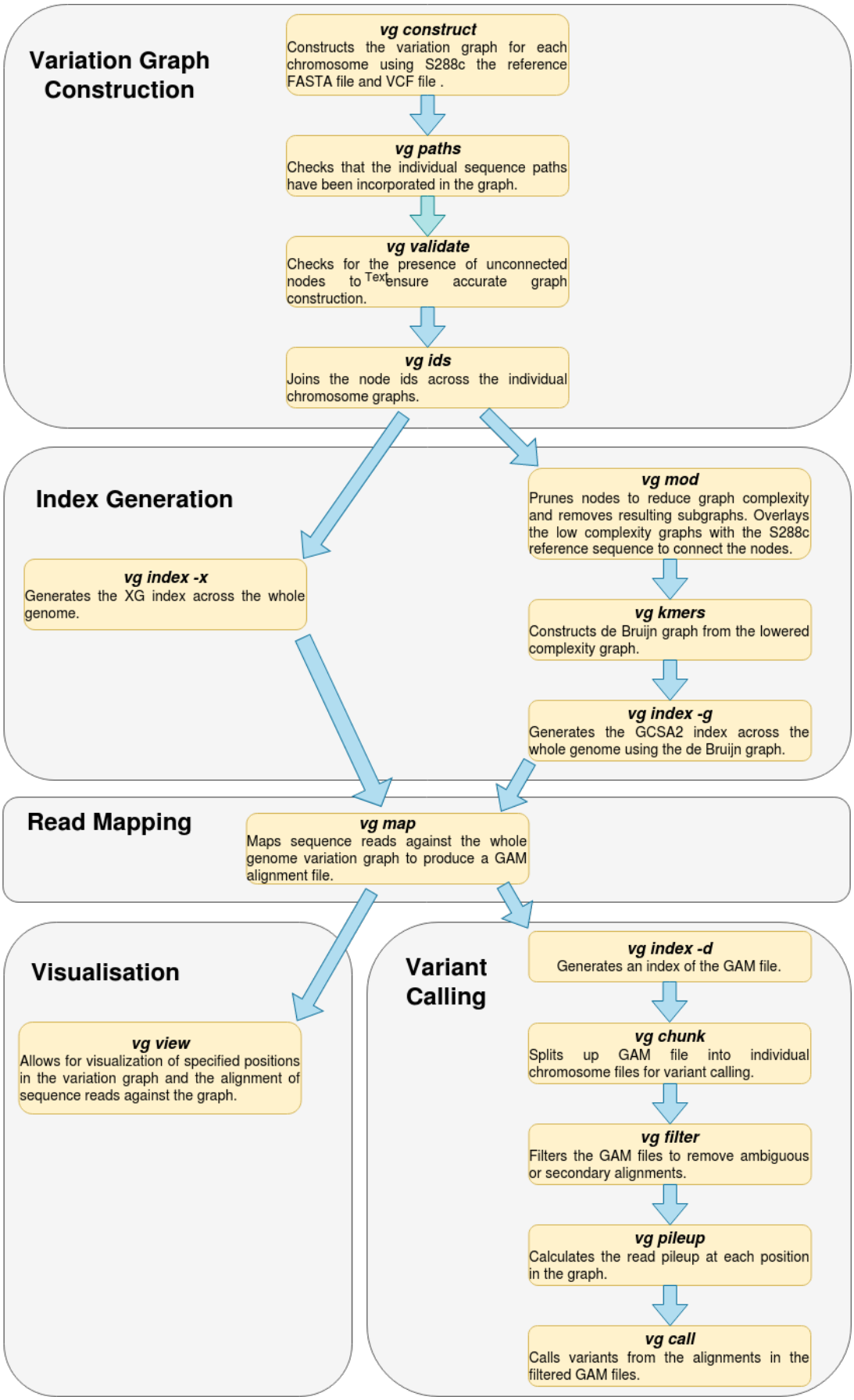


Figure 2.1: ***vg* Workflow.** This figure displays the workflow of the commands utilised for variation graph construction, indexing, read mapping and variant calling in *vg*.

also referred to as a reference/flat graph), in comparison to the traditional linear reference (Garrison *et al.*, 2018, Jones, 2016). Nineteen haploid *S. cerevisiae* strains (see Table 2.1 for information pertaining to the strains) were selected from a study comparing genome content variation between *S. cerevisiae* and its closest wild relative *S. paradoxus* (Bergstrom *et al.*, 2014) as these strains were also part of the *Saccharomyces Resequencing Project* (SGRP), a collaborative project between the Wellcome Sanger Institute and the Institute of Genetics, University of Nottingham (Carter, 2008). The sequence files for the 19 strains, hereafter referred to as the Bergstrom strains, were downloaded from the SGRP website (http://www.mo-seslab.csb.utoronto.ca/sgrp/download.html). The *vg* software was used to generate the Bergstrom variation graph and the S288c reference graph and for the linear pipeline, the *BWA* v0.7.17 (Li and Durbin, 2009) software was used to carry out mapping against the S288c reference genome.

The S288c reference genome was used to generate the reference graph without any variants. Prior to graph construction, sequences pertaining to the mitochondrial chromosome were removed from the reference FASTA file to allow for a fair comparison as the VCF file containing variants from the Bergstrom strains did not include variants for the mitochondrial chromosome. The mitochondrial genome sequence was identified by searching for sequence headers containing the 'chrmt' label and any corresponding sequences were manually removed from the reference FASTA file. The reference graph was constructed with just the reference FASTA file using the *-R* parameter to limit graph construction to each chromosome. The same methods as described in Section 2.2.1 were utilised for graph construction and indexing except that the absence of complex regions within the graph meant that pruning was not necessary when building the GCSA2 index. All 19 strains were mapped against the reference graph and the resulting alignment GAM files were parsed using *jq*. The read sequences and alignment scores for each strain were obtained to evaluate how well the reads mapped against the reference graph.

The variation graph containing the S288c reference genome plus variants from the 19 *S. cerevisiae* strains was created using a multi-sample VCF file downloaded from the SGRP laboratory website. The indexed VCF file and the mitochondrial chromosome-free reference FASTA file were used to construct and index the variation graph, again limiting graph construction to each chromosome. The Bergstrom strains were mapped

Strain	Subpopulation	Strain Habitat	Location	Raw Read Depth
DBVPG1373	Wine/European	Soil	Netherlands	32
DBVPG6044	West African	Bili Wine	West Africa	35
L1374	Wine/European	Wine	Chile	26
SK1	Mosaic	Soil	USA	56
UWOPS83-787.3	Mosaic	*Opuntia spp.*	Bahamas	627
W303	Mosaic	Laboratory	USA	48
Y55	Mosaic	Grapes	France	38
YJM978	Wine/European	Clinical	Italy	32
YPS128	North American	Quercus alba	USA	64
BC187	Wine/European	Wine	USA	23
DBVPG1106	Wine/European	Grapes	Australia	43
DBVPG1788	Wine/European	Soil	Finland	30
DBVPG6765	Wine/European	Litchi fruit	Indonesia	46
L1528	Wine/European	Wine	Chile	49
UWOPS03-461.4	Malaysian	Bertram Palm	Malaysia	32
UWOPS87-2421	Mosaic	*Opuntia spp.*	Hawaii	821
Y12	Sake	Sake	Japan	44
YJM975	Wine/European	Clinical	Italy	33
YJM981	Wine/European	Clinical	Italy	16

Table 2.1: **Bergstrom Strains.** Strain accession ID, subpopulation, habitat, country of origin and estimated sequencing coverage for the 19 strains within the Bergstrom strain set (Bergstrom *et al.*, 2014).

against the variation graph and as for the S288c reference graph, the alignment information was parsed out of the GAM file to evaluate how well the reads mapped when known variants were incorporated within the reference.

In order to carry out read mapping using the traditional linear approach, the S288c mitochondrial chromosome-free reference FASTA file was again used as the reference genome and indexed with *bwa index* before mapping with the *bwa mem* algorithm. Additionally, prior to mapping, the default parameters for scoring alignments in *BWA* were checked against the scoring system employed by *vg* in order to ensure that the alignment scoring was identical, allowing mapped reads from both software to be compared. As the scoring for alignments used by both software was identical, the default parameters

were kept for the read mapping of all 19 strains. The scoring system utilized for read alignments were as follows:

- 1 score for every successful base match.
- 4 score penalty for mismatch.
- 6 score penalty for opening gaps.
- 1 score penalty for each base in gap extension.

The read alignments for each strain were written out to SAM (Sequence Alignment Map) files (Li *et al.*, 2009). The read sequences and the alignment scores were parsed out of the SAM file using *SAMtools* (Li *et al.*, 2009) and the *bioawk* (Li, 2017) tool was used to reverse complement read sequences when flag 16 was present in the SAM file indicating the read was matched to the reverse strand.

The *R* software was utilised to analyse how the alignment scores differed across mapped reads, as identified by read name, from the *vg* reference graph, variation graph and the linear reference genome firstly to compare graphical and linear approaches and secondly to explore whether the addition of known variants to the reference graph had any impact on read mapping. Common reads that mapped, i.e. had an alignment score greater than 0 against both the *vg* reference graph and the linear reference, were identified and duplicate read sequences were removed. The alignment scores were compared to identify reads in which the difference in alignment scores between the two approaches was greater than two. The same method was repeated for reads that mapped against both the *vg* variation graph and the linear reference.

2.2.3 Creating a Pan-Genome Variation Graph

The variation graph constructed in Section 2.2.2 required VCF files obtained by mapping reads to a linear reference, which seemed counter-intuitive. A pan-genome variation graph was therefore created to remove this requirement. Instead, variants were called by mapping haploid *S. cerevisiae* datasets against the reference graph. The called variants were then folded back into the flat graph and this process was repeated in an iterative fashion to incorporate the majority of variants present within the *S. cerevisiae* population. The pan-genome variation graph was used to further evaluate the capacity for read mapping to improve with the continuous incorporation of additional variants.

In the initial step, the reference graph was constructed again from the whole S288c reference genome using the FASTA file containing the complete reference sequence in order to include the mitochondrial chromosome. The reference graph was constructed using the same method as previously outlined and sequence reads from the 19 Bergstrom strains were re-mapped against the reference graph. The read alignment information was parsed from the GAM files using *jq*.

Variants were called from the read alignments against the reference graph in a multi-step process. As the *vg* software cannot handle variant calling on whole genomes, GAM index directories (RocksDB databases that allow for querying of reads) were generated for each strain with *vg index*. The GAM file for each strain was split up into 17 per-chromosome chunks based on the GAM index using the *vg chunk* function. For each strain, any ambiguous or secondary read mappings were filtered out of the GAM file using *vg filter* with the recommended parameter *-r 0.90 -afu -s 2 -o 0 --defray_-ends 999*. The *vg pileup* function was used to calculate pileup for each position in the graph by thresholding based on read support before calling variants with *vg call* to produce per-chromosome VCF files for each strain. The VCF files were normalized using the *Vt* v0.5 software (Tan *et al.*, 2015) to allow for a more unambiguous, unique and concise representation of the variants. First, the *decompose* function in *Vt* was used to break down multi-allelic variants into bi-allelic variants for each chromosome and this was passed into the *decompose blocksub* function to further decompose bi-allelic block substitutions into its constituent SNPs and indels. The decomposed VCF files were normalized in order to make the variants parsimonious (represented in as few nucleotides as possible without reducing the length of the allele to 0) and left-aligned (shift the variant to its left-most position while keeping the length of all its alleles constant). The VCF files were sorted using *VCFtools* v0.1.15 (Danecek *et al.*, 2011) and the *GATK CatVariants* tool was utilized to concatenate the per-chromosome VCF files into a single VCF file for each strain. The strain VCF files were validated using the *GATK ValidateVariants* tool to ensure that they were in the correct format and did not contain errors. The per-strain VCF files were merged into a multi-sample VCF, compressed and indexed to be used to construct the pan-genome variation graph consisting of 20 *S. cerevisiae* strains, including the S288c reference genome.

A further 10 haploid *S. cerevisiae* strains were selected for mapping from the NCYC

database. The strains were chosen such that they were deposited across many decades and from various environments in order to create a diverse population strain set. The following NCYC strains were selected: NCYC78, NCYC97, NCYC430, NCYC672, NCYC1406, NCYC1415, NCYC2517, NCYC2855, NCYC3493 and NCYC3630. The strains were mapped against the reference graph, the pan-genome variation graph and the linear reference genome to compare the efficacy of read mapping across the three methods. The read sequences and alignment scores were parsed out from the GAM and SAM files. Any secondary mapping, chimeric reads and unmapped reads were removed from the SAM files prior to parsing out alignment scores as the *vg* GAM file only contains primary alignments. *R* was utilised to compare the differences in mapping across the three reference structures.

2.2.4 Differences in Mapping Raw and Trimmed Sequence Reads

The *FastQC* v0.11.7 (Andrews, 2010) software was used to carry out quality control analysis of the 19 Bergstrom strains as comparison of read mapping indicated low sequencing quality within the raw sequence reads. Quality control analysis revealed that all of the strains, with the exception of UWOPS83-787.3 and UWOPS87-2421, had poor sequence read quality and indicated the presence of adapter contamination within the sequence reads. The *Trimmomatic* v0.32 software was used to trim the Bergstrom strains in order to remove any adapter sequences. As the Bergstrom strains were sequenced using the Multiplexed PCR-free Illumina sequencing library (Kozarewa *et al.*, 2009), the adapter sequences for both paired-end and multiplex adapters were obtained from the Illumina adapter sequence document (Illumina, 2019) to create a custom adapter FASTA file containing 11 adapter sequences used as input for *Trimmomatic* with the default parameters.

The Bergstrom pan-genome variation graph was re-constructed from the trimmed sequence reads using the same methods as detailed in Section 2.2.3. As with the raw reads, the trimmed sequence reads from the 19 Bergstrom strains and the previously chosen 10 NCYC strains were mapped against the S288c linear reference genome with *BWA*, the S288c reference graph and the Bergstrom variation graph to evaluate how differences in read quality affect the ability of sequence reads to align against a reference structure. The GAM files containing alignments against the reference and variation

graphs were surjected to BAM format using the *vg surject* function to allow for comparison across the different software. A surjected alignment refers to an alignment against a graph-based reference genome that is represented as if it were aligned against the linear reference genome. Conversely, a non-surjected alignment refers to an alignment against the graph itself. The *R* software was used to evaluate the percentage of mapped reads against the different reference structures for all 29 strains. As fewer sequence reads from NCYC78 and NCYC97 were found to align against all three reference structures, with the raw reads aligning better against the *vg* graph-based references, sequence reads that were discarded during trimming were identified using the dedupe.sh script from the *BBMap* v38.05 (Bushnell, 2014) suite, mapped against the three reference structures and the percentage of mapped reads were analysed in *R*.

2.2.5 Comparison of Alignment Scoring by *vg* and *BWA*

In order to be able to select the read mapper that generates optimal alignments for a given dataset, the alignments produced by the various mappers can be compared in different ways. The sensitivity of the read mapping algorithm can be determined easily by the number of sequence reads it is able to align against the reference. Although, greater sensitivity in mapping does not always necessarily mean that the accuracy of the alignment has improved. In terms of the quality of the alignment, most mapping algorithms produce a mapping quality score for each aligned read. The mapping quality score estimates the probability that a read is not aligned to its true location by taking into account the base quality scores at mismatched bases (Li *et al.*, 2008). However, as the mapping quality scores are calculated differently by different mappers, it cannot be used to cross-compare alignment quality. Alignment scores also provide an indication of alignment quality and may be utilised for comparison if the different mapping algorithms employed the same scoring systems. If there is discordance in alignment scoring, an alternative approach may be to calculate the percentage of bases that match exactly against the reference for each aligned read, this is also referred to as the sequence identity. This allows for direct cross-comparison of alignments as if a mapping algorithm produces reads with greater sequence identity, this suggests that the quality of alignment and thus the mapping accuracy has improved.

The alignments generated from mapping the Bergstrom and NCYC strain sets

against both the S288c linear reference genome and the S288c reference graph produced consistently differing alignment scores, in which the scores output from *vg* were usually higher than *BWA* for the same sequence read. This strongly suggested that there were dissimilarities in either the read alignments against both reference structures or the scoring parameters employed by both algorithms. In order to identify any discrepancies in the default scoring parameters utilized by both the *BWA* and *vg* mapping algorithms, the alignment scores for a few selected read alignments from the Australian grape strain, DBVPG1106, were replicated by hand.

The DBVPG1106 SAM file, containing the read alignments from mapping against the linear reference genome, was sorted, indexed and converted into BAM format using *SAMtools*. The BAM file and the S288c FASTA reference file was used as input for the *SAMtools tview* function which allowed for visualisation of the exact read alignment, including any mismatches and indels, to aid in alignment scoring. As there had been several updates to the *vg* software since the initial analyses in Sections 2.2.1-2.2.4, *vg* v.1.9.0 (Miglionico) was installed and the reference graph was re-constructed before re-mapping the Bergstrom strains against the graph. Mapping was carried out twice, both with and without the *'--surject-to-sam'* parameter, to produce the SAM and GAM files respectively as the SAM files surjected from *vg* do not output the alignment scores. The read sequence and alignment scores were parsed from the GAM file into a CSV file and the SAM file was converted into a BAM file using *SAMtools view*. Both the *vg* and linear BAM files were further filtered to only include reads less than 60 bp that mapped to chromosome I of the S288c reference genome to allow for easier inspection and selection of reads for scoring. As the BAM files produced by *vg* did not include the alignment score tags, the individual read sequences and their corresponding alignments scores had to be manually searched from the CSV file.

Visual inspection of the read alignments allowed for the identification of reads that had vastly different alignments between *vg* and *BWA* even when aligning against the same portion of the reference. Utilization of the alignment scores to replicate scoring based on the established scoring parameters also helped determine whether hidden parameters were used in calculating alignment scores and, therefore, whether alignment scores enabled a fair comparison between mappers.

2.2.6 Simulation Study to Identify Factors That May Impact Read Mapping in *vg*

As it is vital to have a thorough understanding of the mapping algorithm employed by *vg* and the limitations of the mapper, a simulation study was carried out to identify the various factors that may prevent a sequence read from aligning against a reference graph. Seven random reference genomes of various sizes were generated using a custom Python script (https://github.com/prithikasritharan/RandomSeqGenerator) that takes a sequence length as input to produce a random DNA sequence of the specified length output in a FASTA genome file. Each FASTA file was indexed using the *SAMtools faidx* function. The lengths of the reference genomes were as follows: 300 bp, 500 bp, 1000 bp, 1500 bp, 2000 bp, 5000 bp and 10000 bp. For the purpose of this study, the lengths of the simulated reference genome sequences were not representative of the size of true genomes to enable closer inspection of read mapping against the reference graphs. A reference graph was constructed for each of the reference genomes and indexed to generate both the GCSA2 and XG indexes. Due to the relatively small sizes of the reference genomes, no pruning was required before generating the GCSA2 index.

For each of the reference graphs, sequence reads were simulated from the reference genome using the *vg sim* function to generate reads ranging from 20 bp to 100 bp, increasing stepwise in size by 10 bp. Across each read length, the following number of reads were simulated for each read set: 100, 200, 400, 600, 800, 1,000, 5,000, 10,000, 50,000, 100,000, 500,000 and 1,000,000. In order to understand how the position and size of variants within a sequence read can affect its mapping ability, various mutations were induced within the sequence reads for each read set. A Python script (https://github.com/prithikasritharan/SequenceReadMutator) was written which takes in each simulated read set and induces the following mutation types (including the percentage of reads containing the mutation within each read set):

- no mutations (10%, used as control).
- mutations at the start (originates from the first base) of the read (20%).
- mutations at the end (originates from the last base) of the read (20%).
- mutations induced at a single, randomly chosen position within the read (20%, referred to as cluster mutations).

- mutations induced at several random positions throughout the read (30%, referred to as point mutations).

The upper limit for the maximum number of mutations induced for a given read, m, was kept constant by setting m to 0.3 x the read length. For each read, the number of mutations to be induced for each mutation type varied between 1 to m.

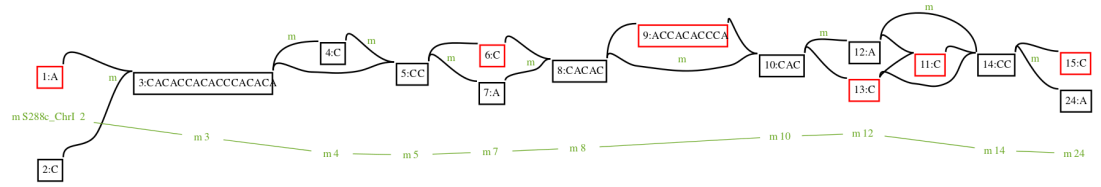
The mutated sequence reads from all the read sets were mapped against their respective reference graphs in *vg* and the *jq* software was used to parse out the mapped reads from the GAM file. The *R* software was used to analyse the percentage of reads that mapped within each mutation type across the different read lengths to understand how the length of the read sequence, the length of the reference genome and the size and position of the variants impact mapping against a reference graph.

2.3 Results

2.3.1 NCYC Strains Variation Graph

The first 16 nodes from chromosome I of the variation graph containing the S288c *S. cerevisiae* reference genome and eight haploid NCYC strains is shown in Figure 2.2a. The genome sequence is contained within nodes in the graph, with the nodes connected via edges and the numbering of the nodes representing the order of the sequence. The S288c reference genome sequence forms the base nodes, shown in black, within the graph. Divergent nodes, highlighted in red, indicate the presence of allelic variants at each specific locus where the sequences differ from that of the reference genome. The order of nodes can be further seen in Figure 2.2b which represents the individual sequence paths within the variation graph for each strain. The graph is able to accurately represent all the possible combinations of sequence variations, such as SNPs, insertions and deletions, present within the genome sequences across the nine strains.

The NCYC variation graph had to be pruned to reduce computational complexity prior to GCSA2 graph index generation, which results in the removal of certain nodes and variant alleles within the graph. The number of nodes and length of sequence for each chromosome graph was observed before and after pruning (see Appendix Table A.1) and the percentage of nodes and sequence retained after pruning can be seen in



(a) Variation Graph

NCYC78	2	3	4	5	7	8		10	12		14	24
NCYC88	2	3	4	5	7	8		10	12		14	24
NCYC93	2	3	4	5	6	8		10	12	11	14	24
NCYC97	2	3	4	5	7	8		10	13	11	14	15
NCYC214	2	3		5	7	8		10	12	11	14	24
NCYC221	2	3		5	7	8	9	10	12	11	14	15
NCYC222	2	3		5	7	8		10	12	11	14	24
NCYC230	1	3		5	7	8		10	12	11	14	24

(b) Individual Sequence Paths

Figure 2.2: **NCYC Strains Variation Graph.** The graph in Figure 2.2a shows the combination of allelic variants present across a population of nine haploid *S. cerevisiae* strains. The red nodes display variant sequences absent from the S288c reference genome and the path labelled in green displays nodes present within the reference genome. Figure 2.2b shows the order of nodes that forms the individual sequence path for each of the NCYC strains from the variation graph in Figure 2.2a. Alternate alleles representing SNPs are shown in blue and insertion sequences in orange.

Figure 2.3. The mitochondrial chromosome had the highest proportion of nodes removed through pruning, 20.82%, resulting in the loss of 8.45% of the allelic variants originally present within the graph. This was followed by chromosome I and X which retained 83.46% and 87.68% of the nodes and 97.59% and 98.36% of the variant sequences, respectively. All of the other chromosomes were able to retain $>$ 99% of the variant alleles and $>$ 92% of the graph nodes.

The ratio of the number of nodes to the sequence length, also known as the N:SL ratio, within the graph can be used to provide a rough indication of the amount of intragenic variation present within the NCYC strain population with a high ratio representing a greater degree of sequence conservation and low ratio representing a greater degree of sequence variation. The N:SL ratio both before and after pruning the varia-

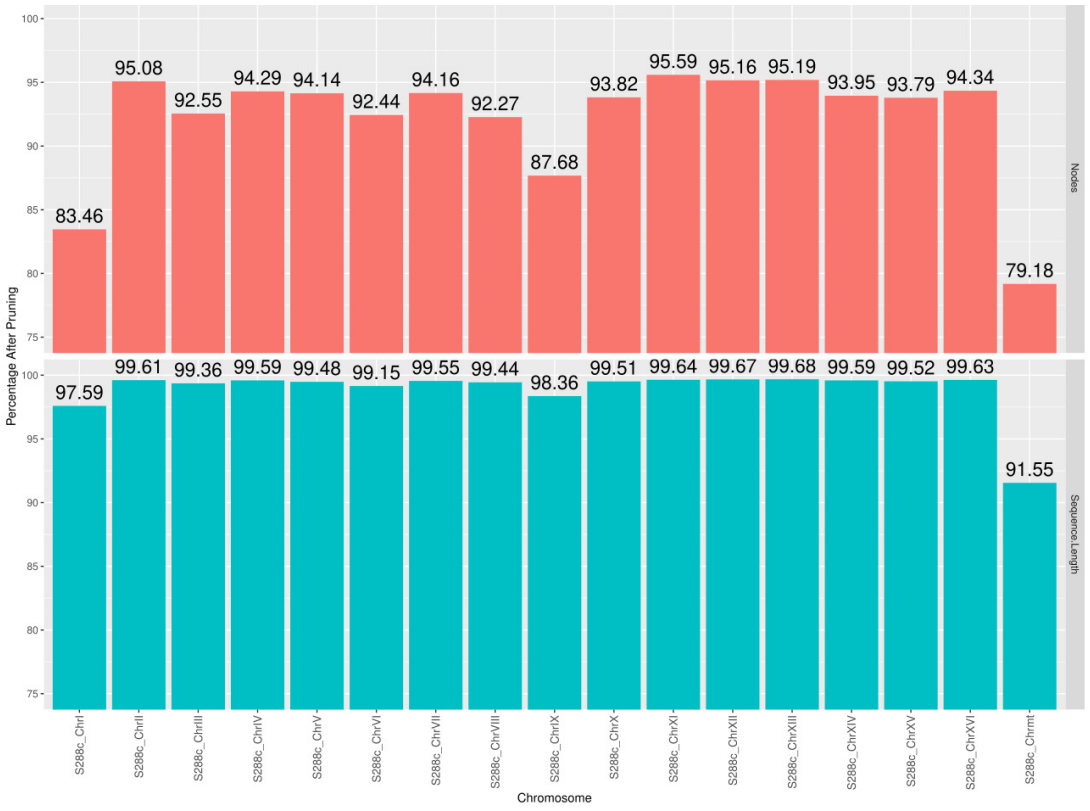


Figure 2.3: **Impact of Pruning on the Variation Graph.** This bar chart shows the percentage of nodes (top) and sequence (bottom) retained within each chromosome of the NCYC variation graph after pruning to remove high complexity regions.

tion graph has been shown in Table 2.2. The mitochondrial chromosome had the lowest N:SL ratio both before (1:7.8) and after (1:9.1) pruning. Both chromosome III and XIII had the highest N:SL ratio before (1:32.2 and 1:33) and after (1:34.6 for both) pruning. The average N:SL ratio across the whole variation graph was 1:26.4 before pruning and 1:28.23 after pruning.

The tetraploid strain, NCYC1006, was aligned against the NCYC variation graph to test whether complex ploidy had an impact on read mapping. The GAM alignment file showed that of the 38,437,506 paired-end sequence reads, 36,753,085 reads were able to align against the variation graph. The percentage of NCYC1006 reads mapped against the graph, 95.6%, was quite high considering the variation graph only contained variant alleles from a small haploid strain population. The qualities of the alignments were studied by comparing the distribution of alignment scores from the mapped reads, as shown in the histogram in Figure 2.4. The frequency of alignment scores in the histogram displays a left-skewed distribution with the majority of the reads mapping

Table 2.2: **Variation Graph Statistics.** Node to sequence length (N:SL) ratio of each of the 17 chromosome graphs before and after pruning.

Chromosome	N:SL Ratio Before Pruning	N:SL Ratio After Pruning
ChrI	1:14.4	1:16.8
ChrII	1:26.7	1:27.9
ChrIII	1:32.2	1:34.6
ChrIV	1:29.4	1:31.0
ChrV	1:24.0	1:25.3
ChrVI	1:19.9	1:21.3
ChrVII	1:28.7	1:30.3
ChrVIII	1:28.0	1:30.2
ChrIX	1:14.4	1:16.2
ChrX	1:25.6	1:27.2
ChrXI	1:27.1	1:28.3
ChrXII	1:30.2	1:31.6
ChrXIII	1:33.0	1:34.6
ChrXIV	1:31.8	1:33.7
ChrXV	1:27.4	1:29.1
ChrXVI	1:30.6	1:32.4
Chrmt	1:7.8	1:9.1

with an alignment score greater than 100. The maximum alignment score from the mapped reads was 118 whilst the minimum score was 19.

2.3.2 Comparison of Read Mapping in *vg* and *BWA*

The efficacy of mapping sequence reads from the 19 Bergstrom strains against the S288c linear reference genome, the S288c reference graph and the pan-genome Bergstrom variation graph was evaluated by comparing the number of sequence reads that were able to align against each reference structure, as shown in Figure 2.5 and Table 2.3. Greater proportions of reads were found to align against the *vg* graph-based reference genomes in comparison to the linear reference genome, even in the absence of variants within the graph. Additionally, the Bergstrom variation graph had the highest percentage of read mapping across all 19 strains. An increase of 0.53 - 0.9% per strain was seen in read mapping from the reference graph to the variation graph. The percentage of reads

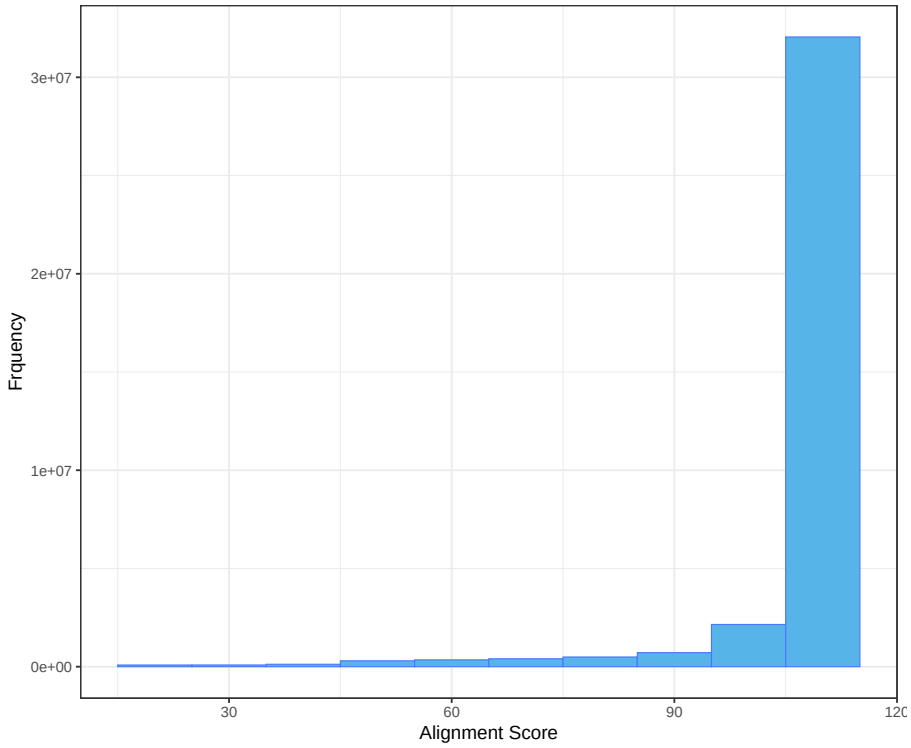


Figure 2.4: **NCYC1006 Alignment Score Histogram.** This histogram shows the distribution of alignment scores from the NCYC1006 sequence reads that mapped against the NCYC variation graph containing nine haploid strains.

mapping against the reference graph was found to be 0.75 - 25.3% higher per strain than the linear reference, with UWOPS87-2421 showing the least difference in mapping and DBVPG1788 showing the greatest difference. The average percentage of mapped reads across all 19 strains for the Bergstrom variation graph was 75%, with 74.3% for the reference graph and 61.2% for the linear reference genome. The *vg* graph-based references aligned the least proportion of reads for the clinical strain, YJM981, in which nearly half of the sequence reads were unable to align, whilst *BWA* performed worst at aligning DBVPG1788 from which over 62% of the sequence reads were unable to align against the reference.

As both *vg* and *BWA* utilised similar scoring systems for read alignments, the alignment scores from reads that mapped against the variation graph, reference graph and the linear reference genome were compared to evaluate the differences in the alignment quality for each reference structure. Figure 2.6 shows the percentages of reads that had similar alignment scores i.e. where the differences in alignment scores were less than two, reads that had greater alignment scores in *vg* than in *BWA* and reads that had greater

Table 2.3: **Comparison of Reads Mapped Against Graph-based and Linear References.** Percentage of reads mapped from each strain against the S288c reference graph, the variation graph containing variants from the 19 strains and the linear reference genome.

Bergstrom Strain	**Reference Graph**	**Variation Graph**	**Total *vg* Reads**	**Linear Reference**	**Total *BWA* Reads**
UWOPS83-787.3	90.79	**91.32**	76271936	89.97	76488926
UWOPS87-2421	89.31	**89.91**	99868770	88.56	100084623
YPS128	83.21	**83.86**	7246910	75.11	7262829
DBVPG6044	78.24	**79.03**	3958528	71.43	3970267
DBVPG1373	79.51	**80.20**	3622082	68.68	3630949
DBVPG6765	77.89	**78.53**	5209684	66.16	5219548
DBVPG1106	77.65	**78.21**	4793632	65.89	4808220
Y55	75.89	**76.64**	4238496	62.46	4249369
W303	75.40	**75.97**	5354174	60.91	5361842
UWOPS03-461.4	73.96	**74.57**	3643564	57.94	3652283
SK1	72.68	**73.58**	6327960	57.55	6347192
YJM975	72.36	**72.92**	3699532	54.68	3705351
Y12	71.75	**72.51**	4962416	54.23	4975500
YJM981	54.82	**55.62**	1821806	53.94	1841504
L1374	71.79	**72.49**	2892448	53.84	2896475
L1528	70.70	**71.56**	5515702	53.09	5523403
BC187	69.20	**70.13**	2626978	51.99	2634591
YJM978	64.37	**65.04**	3598544	39.68	3602326
DBVPG1788	62.91	**63.80**	3380390	37.62	3386194

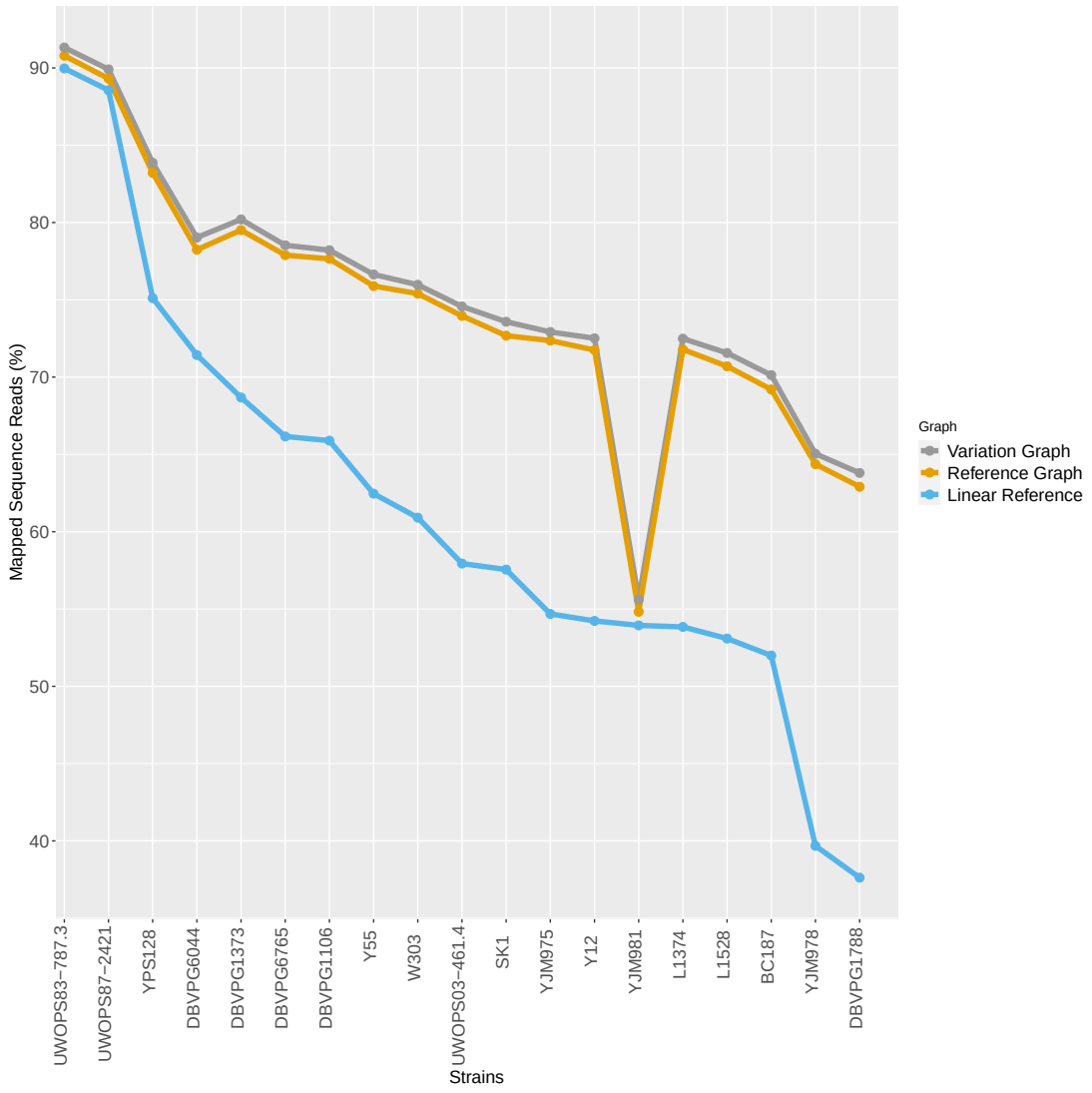


Figure 2.5: **Comparison of Read Mapping in *vg* and *BWA*.** This figure shows the percentage of reads from each of the 19 Bergstrom strains that aligned against the S288c linear reference genome (blue), S288c reference graph (orange) and the Bergstrom variation graph (grey).

alignment scores in *BWA* than in *vg*, when comparing alignments from the variation graph against the linear reference and the reference graph against the linear reference. Approximately 94-98% of the reads in 18 of the Bergstrom strains were able to map with higher alignment scores in *vg* in comparison to *BWA* for both the reference and variation graph, with 0.1-4.4% of reads aligning with greater alignment scores against the variation graph than the reference graph. However, only 0.7-3.1% of the reads had similar alignment scores between the variation graph and the linear reference whilst 1-3.4% of reads had similar scores between the reference graph and the linear reference. A similar percentage of reads were found to align with greater alignment scores in *BWA*

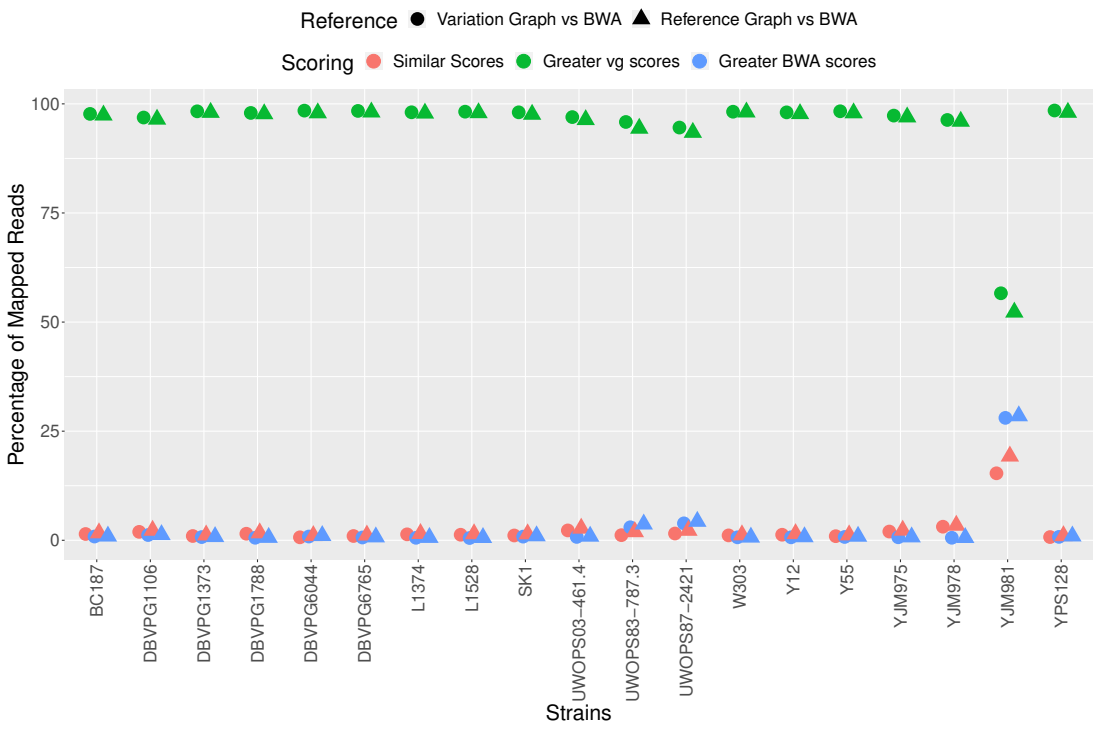


Figure 2.6: **Comparison of Alignment Scores from *vg* and *BWA*.** This figure shows the percentages of common reads that mapped with similar alignments scores i.e. where the difference in alignment scores < 2 (pink), greater alignment scores in *vg* (green) and greater alignment scores in *BWA* (blue) when comparing the variation graph against *BWA* (circle) and the reference graph against *BWA* (triangle).

than *vg*, 0.5-3.9% and 0.6-4.3% of reads when comparing against the variation and reference graph, respectively. The only exception was the strain YJM981 which mapped very poorly in *vg*, such that only 56.6% of the reads that mapped to the variation graph and 52.2% of reads that mapped to the reference graph had higher alignment scores. In addition, there was an increased proportion of reads that aligned with greater alignment scores in *BWA* than reads that aligned with similar scores. Around 28% of the reads aligned with better scores in *BWA* in comparison to both graph-based references whilst 15.4% of reads mapped with similar scores against the reference graph and 19.3% of reads against the variation graph.

The alignment scores of common reads from the 19 Bergstrom strains that mapped against the reference and variation graph were also compared in a similar manner as shown in Figure 2.7. The strain, UWOPS83-787.3, had the highest proportion of reads, 91.9%, that mapped against both the reference and variation graph whilst YJM981 had

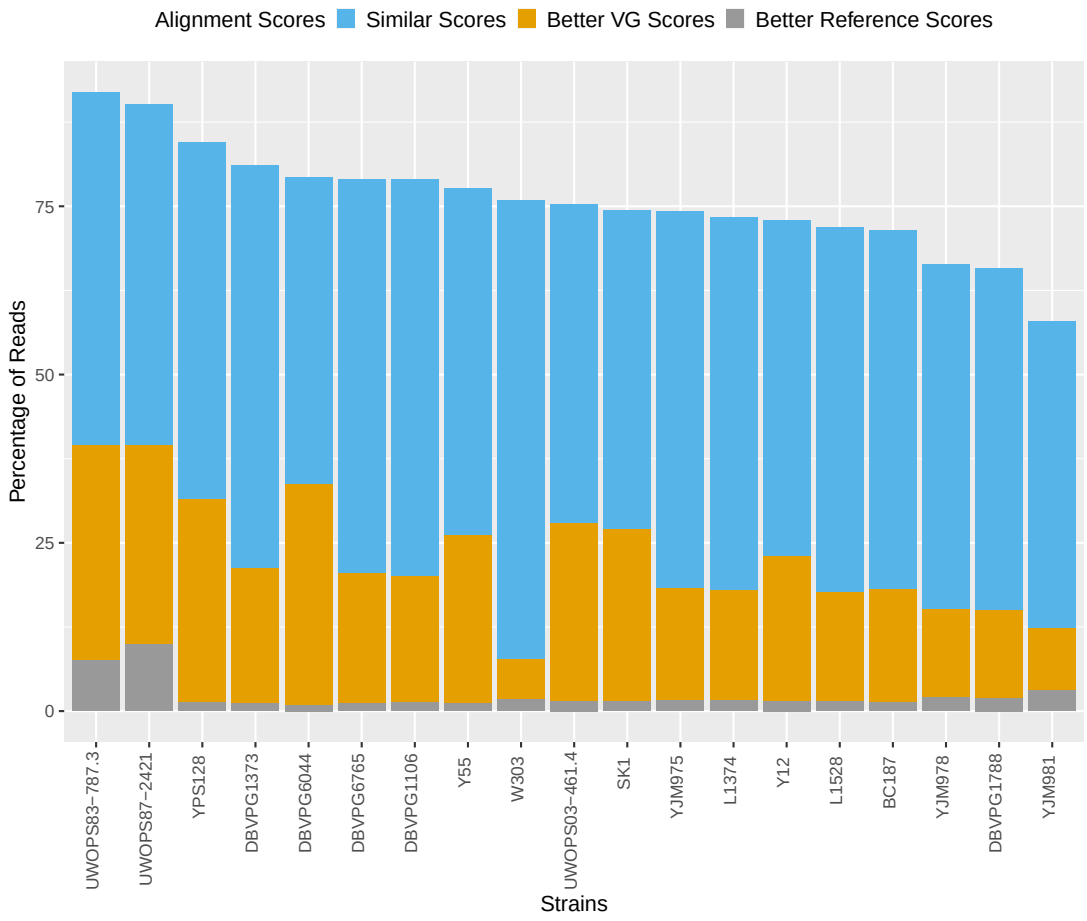


Figure 2.7: **Comparison of Alignment Scores Between the Reference and Variation Graph.** This figure compares the alignment scores of common reads that aligned with similar alignment scores (blue), greater alignment scores against the reference graph (pink) and greater alignment scores against the variation graph (green).

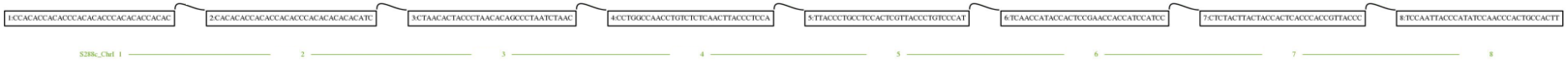
the lowest proportion of common reads, 57.9%. As expected, the majority of reads were found to align with similar alignment scores against both the reference and variation graph for all strains. The mosaic laboratory strain, W303, was found to have the greatest percentage of reads, 68.2%, that aligned with similar scores across both graph-based references. However, between 6-32% of the reads across all strains were also found to align with higher scores when aligning against the variation graph, with W303 containing the lowest proportion of common reads that had higher scoring alignments against the variation graph. In comparison, only 0.9-3.2% of reads had greater scoring alignments against the reference graph except UWOPS83-787.3 and UWOPS87-2421 which had 7.6% and 10.2% of reads, respectively. These findings further supported the idea that alignment quality improves when aligning against the variation graph in comparison to the reference graph and linear reference genome.

2.3.3 Pan-Genome Variation Graph

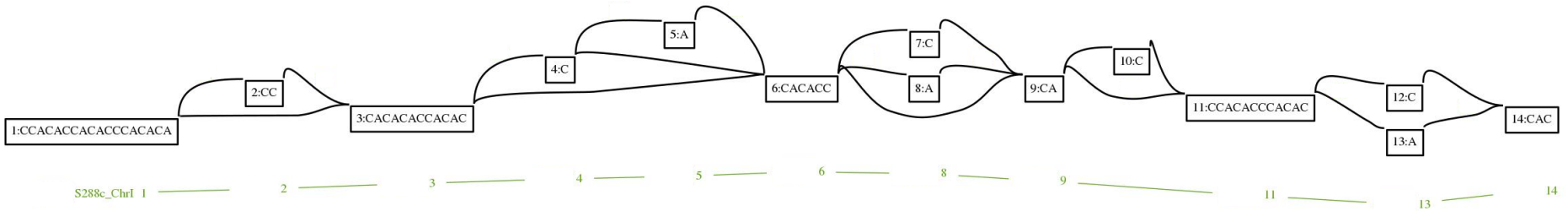
Figure 2.8 exhibits the graph structure of the reference graph in comparison to the Bergstrom variation graph, the latter re-constructed from variants called against the reference graph in order to incorporate the mitochondrial genome, and the initial NCYC variation graph. The reference graph depicted in Figure 2.8a only consists of the S288c genome hence contains nodes with long stretches of sequences connected by straight edges. The Bergstrom variation graph (see Appendix Table A.1 for the graph statistics) contains the genomes of 20 strains hence there is a greater level of sequence diversity which is represented by divergent nodes, as seen in Figure 2.8b. There is a clear difference between the Bergstrom variation graph and the NCYC variation graph (Figure 2.8c) as certain variants that are present in one population are not found in the other, such as the SNP in the first node of the NCYC variation graph which indicates a C $\rightarrow$ A base substitution in one or more genomes within the NCYC strain population that is absent in the Bergstrom strain population despite having a greater number of genomes. These differences demonstrate that the structure of the variation graph is essentially determined by the genomes contained within it.

As the mitochondrial genome was previously estimated to contain the greatest level of sequence diversity, the differences in read mapping when the mitochondrial chromosome was included and excluded was examined by comparing the proportion of reads that were able to align against the S288c reference graph, as shown in Figure 2.9. Approximately 2-5% fewer reads were able to map when the mitochondrial chromosome was absent. The strain UWOPS87-2421 had the greatest increase, 5.57%, in the percentage of mapped reads whilst YJM981, which mapped poorly in general had the least improvement, 2.19%, in read mapping.

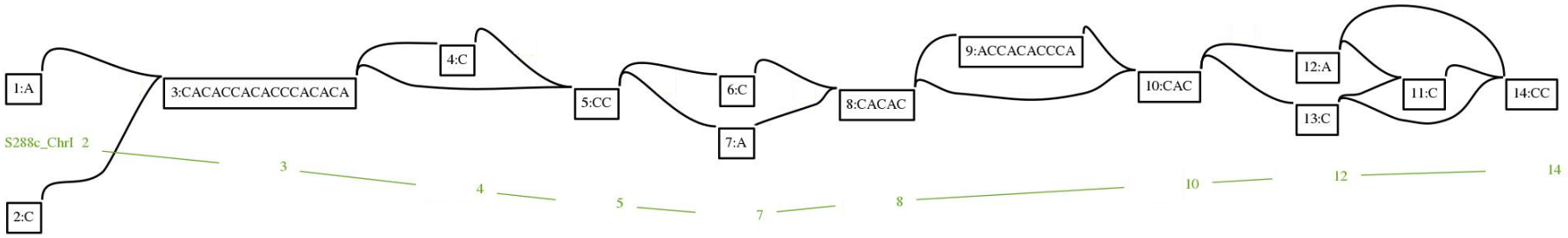
The alignment of sequence reads from a new set of 10 NCYC *S. cerevisiae* strains against the three reference structures was again compared in Table 2.4 to further evaluate how well reads align when using high quality sequence reads. The results in the table confirm previously seen results as there was an increased difference between the proportion of reads that mapped against both the graph-based references and linear reference. Around 5.6-10% fewer reads were able to map against the linear reference than the reference graph and 5.9-10.1% fewer reads than the variation graph. The variation graph had the highest proportion of reads aligned for each strain followed by the



(a) Reference Graph



(b) Bergstrom Variation Graph



(c) NCYC Variation Graph (Figure 2.2a)

Figure 2.8: **Differences in Graph Structure.** This figure demonstrates the differences in graph structure between (a) the S288c reference graph, (b) the Bergstrom variation graph which consists of S288c plus variants from 19 *S. cerevisiae* strains and (c) the initial NCYC variation graph which consists of S288c plus variants from 8 NCYC strains. The first few nodes from chromosome I of all the graphs are displayed above. The edges and paths labelled in green represent the order of nodes present in the S288c reference genome.

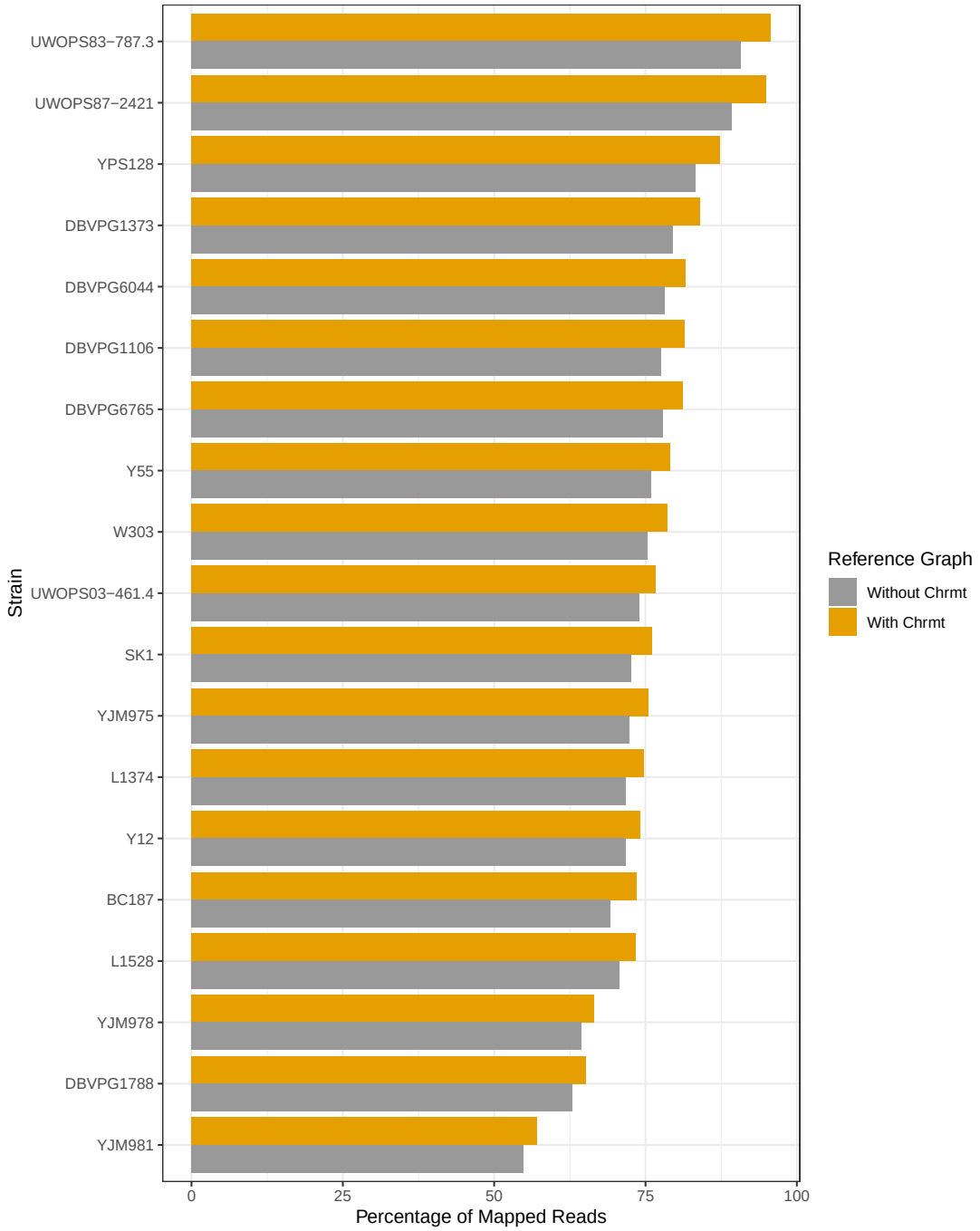


Figure 2.9: **Read Mapping With & Without Mitochondrial Chromosome.** This bar chart shows the percentage of sequence reads that aligned against the S288c reference graph both with and without the mitochondrial chromosome. The orange bar shows the percentage of reads that aligned against the complete reference graph and the grey bar shows the percentage of reads that mapped against the graph without the mitochondrial chromosome.

reference graph which mapped 0.1-0.3% fewer reads. The maximum alignment scores from *BWA* were observed to be exactly 10 less than *vg* for all strains. Additionally, the maximum alignment scores for the majority of NCYC strains were also found to be much higher across all three reference structures than the Bergstrom strains (118). The clinical strain NCYC3493 had the highest alignment score of 261 in *vg*, with 81.5% and 83.3% of the reads mapping with alignment scores greater than or equal to 200 against the reference and variation graph. 98% of reads from NCYC3630, a MATα derivative of the Bergstrom strain Y12, were able to map against the reference graph whereas only 74% of Y12-derived reads mapped.

Table 2.4: **NCYC Reads Mapped Against Graph-based and Linear References.** The number and percentage of sequence reads from 10 NCYC haploid *S. cerevisiae* strains that mapped against the reference graph, pan-genome Bergstrom variation graph and linear reference. Both the minimum and maximum alignment scores from mapping with *vg* and *BWA* for each strain are displayed below.

Strain	**Reference Graph**	**Variation Graph**	***vg* Alignment Scores**		**Linear Reference**	***BWA* Alignment Scores**	
			Min	**Max**		**Min**	**Max**
NCYC78	97.80	**97.95**	19	111	91.65	19	101
NCYC97	95.16	**95.47**	19	111	89.51	19	101
NCYC430	98.06	**98.15**	19	135	91.81	19	125
NCYC672	96.35	**96.57**	19	135	87.86	19	125
NCYC1406	97.72	**97.82**	19	135	89.73	19	125
NCYC1415	97.43	**97.59**	19	135	88.48	19	125
NCYC2517	97.31	**97.41**	19	135	88.77	19	125
NCYC2855	97.46	**97.62**	19	135	91.75	19	125
NCYC3493	98.00	**98.10**	19	261	91.33	19	251
NCYC3630	98.18	**98.34**	19	135	88.20	19	125

2.3.4 Raw and Trimmed Sequence Read Mapping

The percentage of both raw and trimmed sequence reads from the Bergstrom strains that aligned against the three reference structures are shown in Figure 2.10. The trimmed read mapping was found to follow the same pattern as the raw reads for all 19 strains,

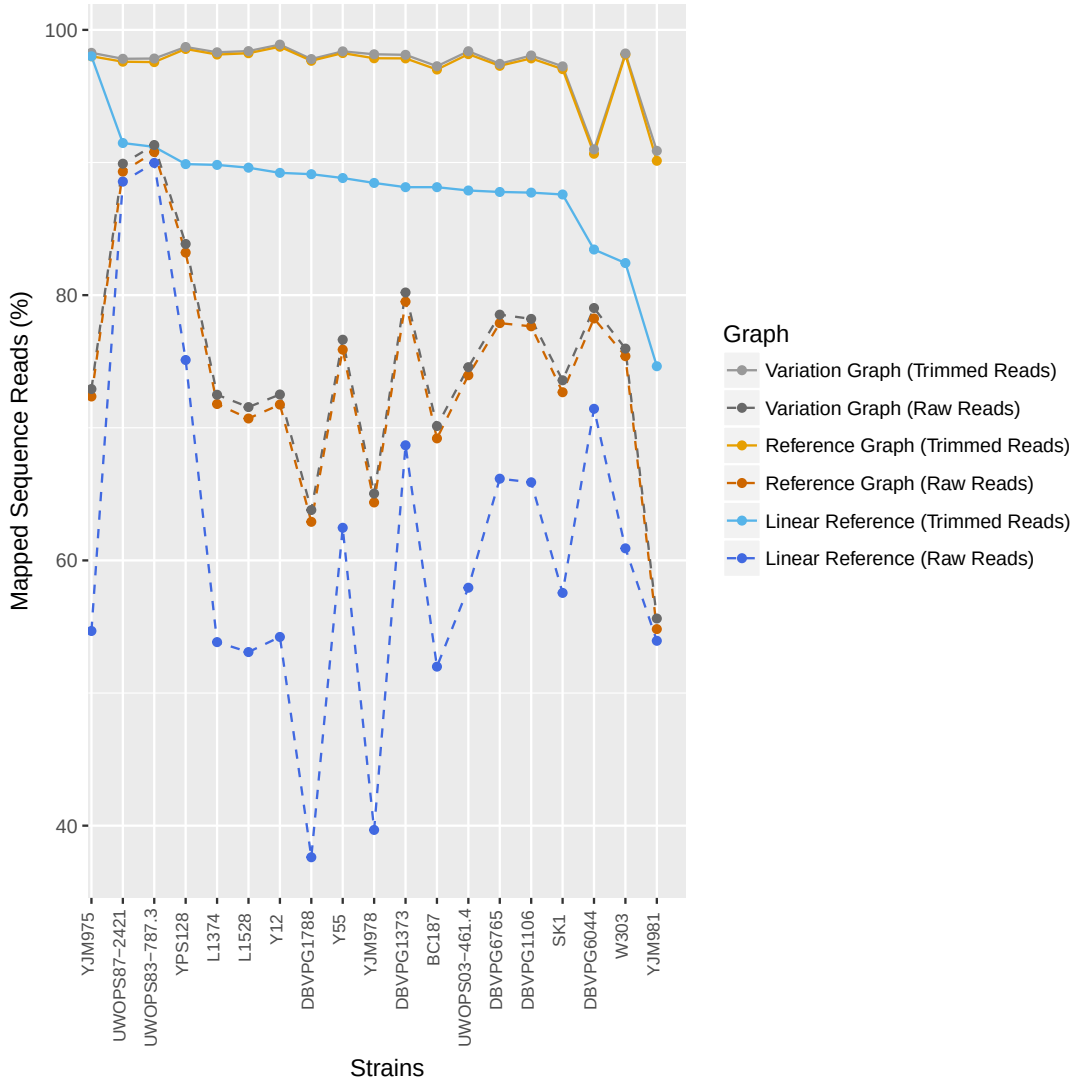


Figure 2.10: **Bergstrom Strains Raw and Trimmed Sequence Read Mapping.** This figure shows the percentage of raw (shown by the dotted lines) and trimmed sequence reads (solid lines) from each strain that aligned against the Bergstrom variation graph (grey lines), the S288c reference graph (orange lines) and the S288c linear reference genome (blue lines).

in which the greatest percentage of sequence reads aligned against the variation graph, as expected. The percentage difference in mapping between the variation graph and the reference graph was found to be 0.53-0.93% and 0.08-0.7% and the difference between the reference graph and the linear reference genome was 0.75-25.29% and 0-15% for the raw and trimmed reads, respectively.

Due to the variable read quality of the Bergstrom strains, trimming discarded over 40% of the sequence reads for 10 strains, with this reaching 80% for the strain YJM981.

Table 2.5: **Mapping Average of the Bergstrom Strains.** Average percentage of raw and trimmed reads that mapped across all three reference structures.

Reference	**Raw Reads (%)**	**Trimmed Reads (%)**
Bergstrom Variation Graph	75	97.3
S288c Reference Graph	74.3	97.1
S288c Linear Reference	61.2	88

However, a greater proportion of the trimmed sequence reads were able to align across all three references compared to the raw reads, with the percentage of raw reads that aligned against the variation graph being consistently lower, or equal for UWOPS83-787.3, than the percentage of trimmed reads that aligned against the linear reference. This is further supported by Table 2.5 which shows an over 20% increase in the average percentage of mapped reads across the references.

As trimming the Bergstrom strains resulted in a considerable reduction in the number of sequence reads, the analysis was again repeated with the 10 NCYC strains. Figure 2.11 displays the percentage of raw and trimmed sequence reads that aligned against the three reference structures. The differences in mapping between the Bergstrom variation graph and the S288c reference graph was 0.09-0.38% and 0.09-0.31% and the difference between the reference graph and the linear reference genome was 5.8-10.2% and 5.65-9.98% for the raw and trimmed reads, respectively. For alignments against the linear reference, as with the Bergstrom strains, trimming was found to slightly increase the proportion of sequence reads that were able to align. However, the opposite effect was observed for both the reference and variation graph in which trimming was found to decrease the proportion of mapped reads for all strains except NCYC78 and NCYC97. In order to understand the exception to the trend displayed by NCYC78 and NCYC97, sequence reads that failed trimming from both strains were realigned across the three reference structures. While none of the discarded reads from both strains were able to align against the linear reference genome, approximately 50% of the reads were able to map against the reference and variation graphs.

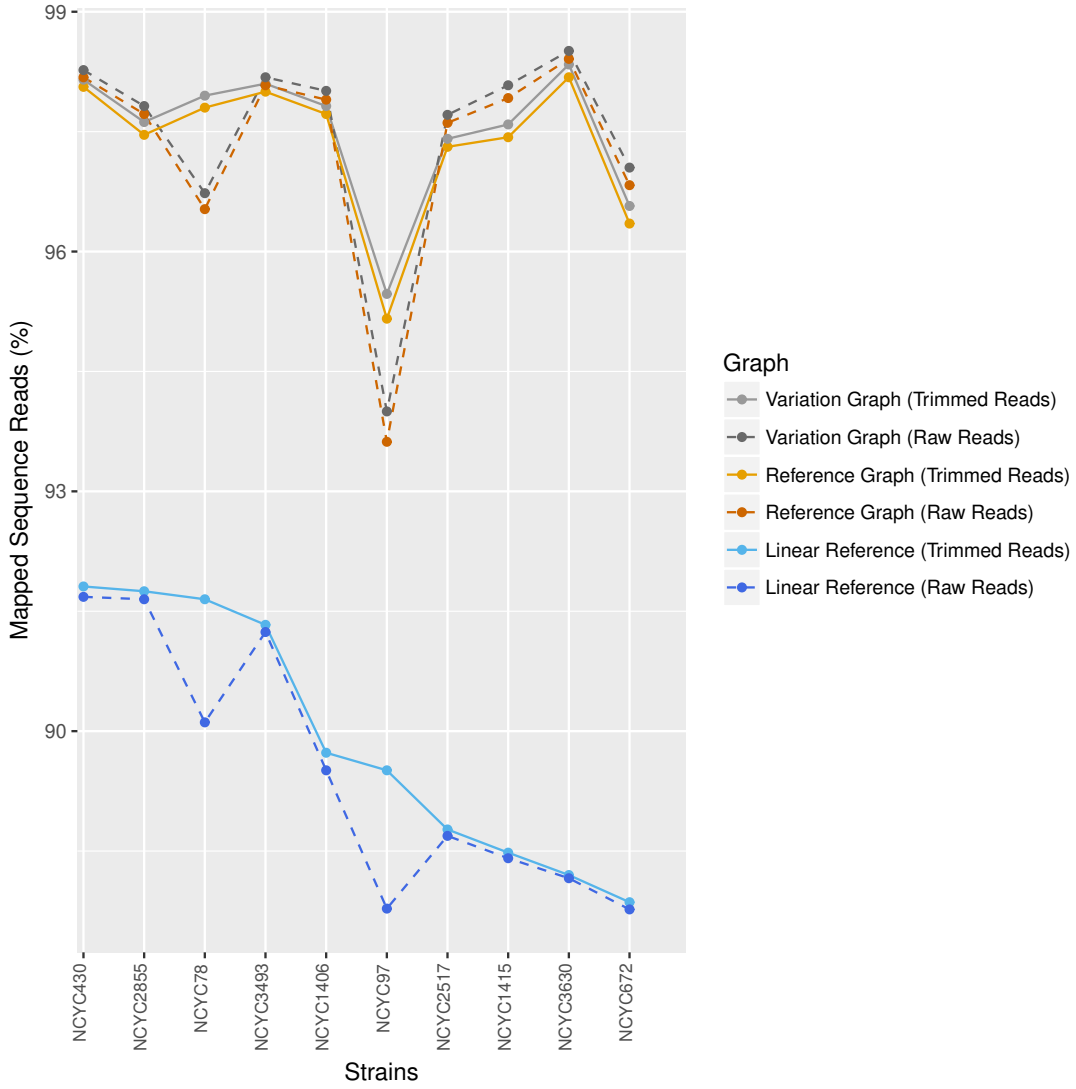


Figure 2.11: **Raw and Trimmed Sequence Read Mapping of the NCYC strains.** This figure shows the percentage of raw (shown by the dotted lines) and trimmed sequence reads (solid lines) that aligned against the Bergstrom variation graph (grey lines), the S288c reference graph (orange lines) and the S288c linear reference genome (blue lines).

2.3.5 Comparison of *vg* and *BWA* Alignment Scores

The findings in Table 2.4, where maximum alignment values were observed to be 10 units higher in *vg* than in *BWA*, indicated the likelihood of hidden parameters in scoring that may bias alignment scores in favour of *vg*. Thus, alignment scores for a selection of sequence reads from the strain DBVPG1106 were re-created to further investigate the scoring systems utilised by *vg* and *BWA*. By default, the *vg* mapping algorithm was found to add a full-length alignment bonus score of five on mapped reads resulting in

a 10 point addition on the alignment score in proper paired-end read mapping. This increase in alignment score can make read alignments from *vg* seem more optimal when compared against other mappers. However, in more recent software versions, *vg* has provided the '--drop-full-l-bonus' option to remove this bonus, thereby enabling fairer comparisons.

The two mappers were also found to vary in how gap scoring was penalised when insertions or deletions were present within the read sequence. Both mappers deduct a penalty score of six for opening gaps and a further penalty score of one for every base gap extension. *BWA*, however, also penalises the first base in the insertion/deletion sequence with a gap extension penalty in addition to the gap opening penalty. Figure 2.12 shows the alignment of a sequence read containing the insertion sequence TTTT against the S288c reference genome and the calculations for the alignment score assigned by each mapper. *BWA* applies the gap extension penalty to all four bases instead of three as with *vg* and *vg* has additionally applied a 10 point bonus for full-length alignment.

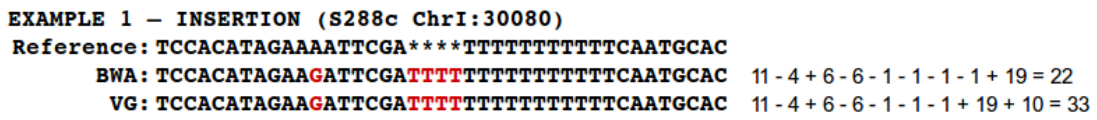


Figure 2.12: **Gap Scoring in *vg* and *BWA*.** The figure above displays an example alignment of a sequence read to illustrate how gaps are scored by both mappers. Both the insertion sequence (TTTT) and any mismatched bases are shown in red. The calculations for obtaining the alignment scores are shown next to the alignment.

Alignment against a position containing the base N, due to the uncertainty of the base call at that position, either within the sequence read or the reference genome was also found to be treated differently by both mappers. As shown in Figure 2.13, the alignment against the position with a base skip in the read is given a penalty score of one by *BWA* during alignment. However, the *vg* mapper does not apply any penalty score to the alignment and tends to ignore the base skip altogether during scoring.

The alignment scores for a small number of reads aligned by *BWA* could not be reproduced and were dissonant to the actual alignments as based on the CIGAR strings. The CIGAR string information found within BAM files is a succinct representation of the read alignment against a reference.

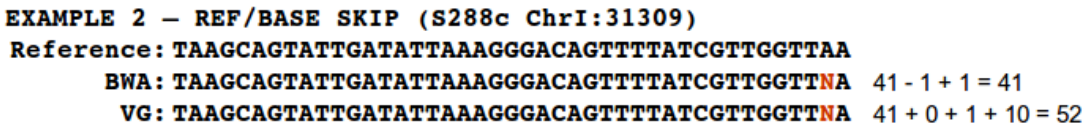


Figure 2.13: **Scoring of Reference/Base Skips in *vg* and *BWA*.** The figure above displays the alignment of a sequence read containing a base skip, represented by the base N shown in red.

2.3.6 Factors that Impact Sequence Read Mapping

In order to further understand the types of variants within a sequence read that can prevent mapping, the percentage of mapped sequence reads were observed for each mutation type. 100% of the sequence reads containing no mutations, mutations at the start of the read and mutations at the end of the read were able to map across all seven reference graphs. This finding demonstrated that mutations at either the start or end of a sequence read, regardless of the number of mutations, do not prevent read alignment against the reference.

Figure 2.14 shows the percentage of reads containing clustered mutations that were able to map. For all of the reference graphs, the percentage of mapped reads at 20 bp read length was quite low, with only 38% mapping against the 10,000 bp reference graph and gradually increasing to 73% mapping against the 300 bp reference graph. There is a steep increase in mapping by 30 bp read length before reaching 100% mapping at 40 bp length suggesting that sequence reads are less tolerant of variants at shorter read lengths. Point mutations were found to have the greatest effect on mapping ability, as seen in Figure 2.15, as point mutations could prevent read mapping regardless of read length. As with the cluster mutations, the percentage of mapped reads increases as the read length increases, reinforcing the idea that tolerance to mutation increases with read length. However, unlike other mutation types, even at 100 bp read length, mapping does not reach 100%. For both the cluster and point mutations, as the size of the reference graph increased for each read length, the percentage of mapped reads was also found to decrease. This is potentially due to the sequence read spanning a greater portion of the genome itself when the reference genome is smaller allowing for less ambiguity in alignment and an increased tolerance to mutations i.e. a 100 bp read spans a third of a 300 bp reference and is therefore highly likely to align even with mutations whereas it spans a one hundredth of a 10000 bp reference, thereby increasing

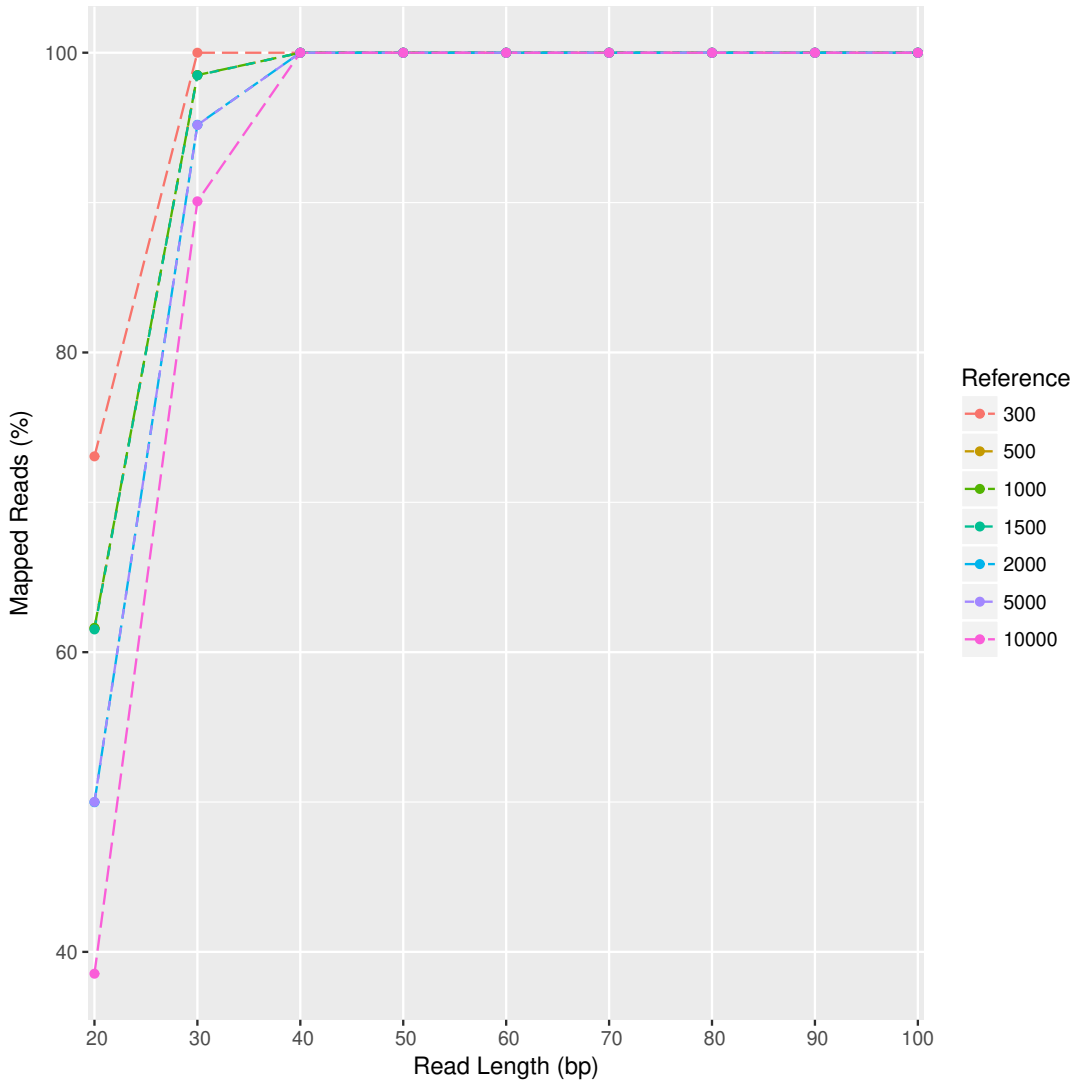


Figure 2.14: **The Effect of Clustered Mutations on Read Mapping.** The line graphs demonstrate the percentage of reads that are able to map when containing clustered mutations for each read length across the seven reference genomes. The percentage of reads that aligned against the 500bp, 1,000bp and 1,500bp reference graphs and the 2,000bp and 5,000bp reference graphs were found to be extremely similar and therefore, the corresponding lines are not distinguished in the graph.

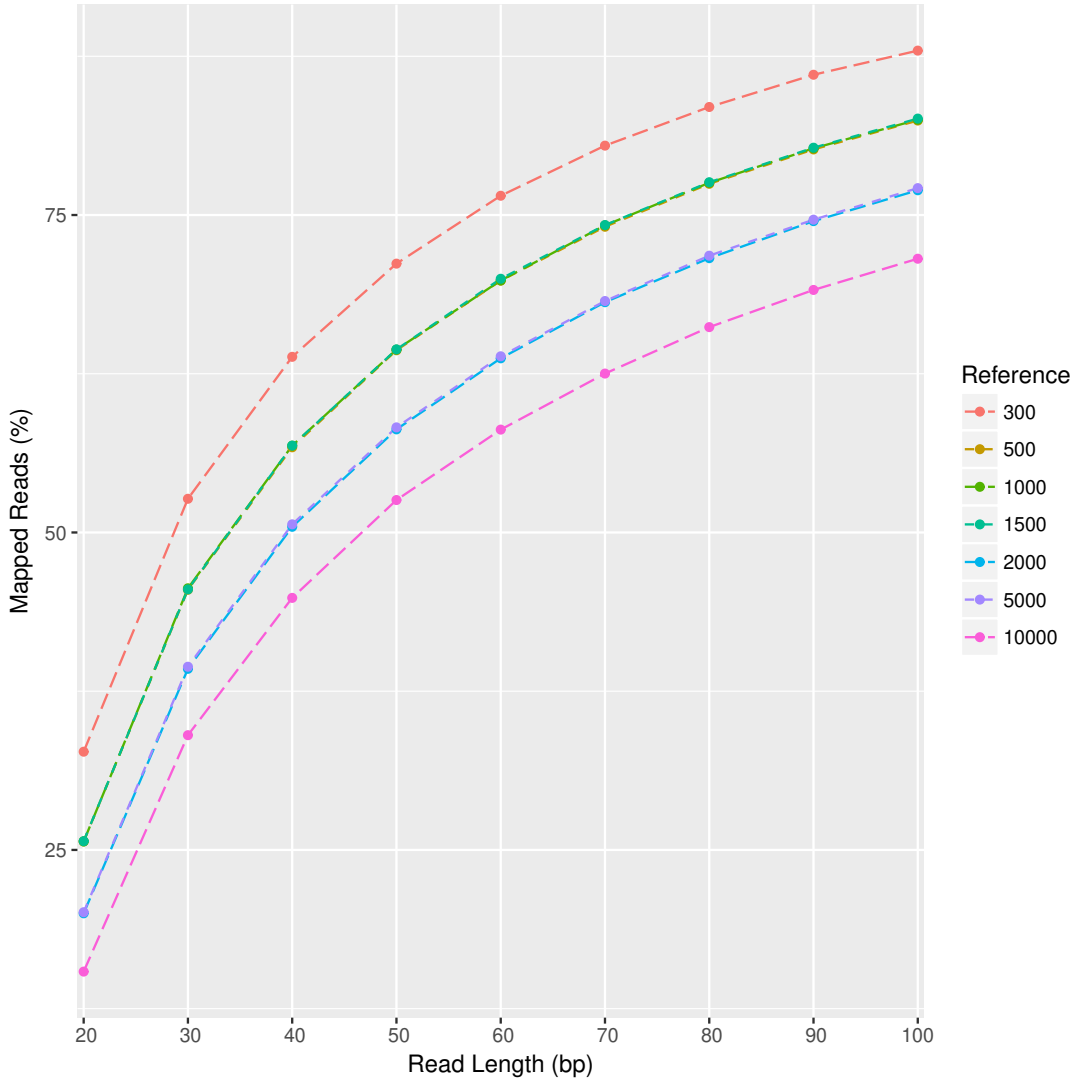


Figure 2.15: **The Effect of Point Mutations on Read Mapping.** The line graphs demonstrate the percentage of reads that are able to map when containing point mutations (mutations induced at several random positions throughout the read) for each read length across the seven reference genomes. The percentage of reads that aligned against the 500bp, 1,000bp and 1,500bp reference graphs and the 2,000bp and 5,000bp reference graphs were found to be extremely similar and therefore, the corresponding lines are not distinguished in the graph.

the difficulty of alignment.

2.4 Discussion

The NCYC variation graph was constructed in order to gain a better understanding of methods utilised for graph construction, read mapping and variant calling using the *vg* software, which was the only available publicly released graph genome software at the time of analysis. Construction of the variation graph was a straight-forward process that was not computationally intensive so long as individual graphs were constructed for each chromosomal region. A major notable benefit provided by the *vg* toolkit was that it allowed for visualisation of both the variation graph and read alignments against the graph. This enabled a thorough understanding of how different variant alleles within a population can be represented in the graph. Although, due to the vast memory requirements necessary for whole-genome graph visualisation, only small regions of the genome can be selected for visualisation providing a snapshot view of the variation graph.

The *vg* subprocess that required the most memory usage was the generation of the GCSA2 index, therefore necessitating pruning of the variation graph to remove regions of high complexity prior to indexing. Any regions that were removed through pruning were overlayed with the S288c reference genome sequence to ensure that the resulting graph remained connected. Initial attempts at indexing the graph without pruning resulted in the process being terminated after running out of memory when run on the NCYC server, which had 516 GB of available memory. However, it should be mentioned that memory requirements for indexing, read mapping and variant calling have all been reduced vastly in later versions of *vg*.

Observing changes in the composition of the variation graph was essential for understanding how variants were discarded during pruning. The mitochondrial chromosome showed the greatest post-pruning loss in both the number of nodes and variant sequences which strongly suggested that it had the highest graph complexity due to it containing the greatest degree of sequence variation amongst all of the chromosomes. Conversely, chromosome XI had the least proportion of nodes pruned. The node to sequence length ratio was also used to further estimate the degree of intragenic variation as highly poly-

morphic regions of the genome contain more variants hence the variation graph is likely to consist of an increased number of nodes with fewer sequences per node, resulting in a low N:SL ratio. The N:SL ratio indicated that the mitochondrial genome, chromosomes I and XI were the least conserved regions amongst the NCYC strain population whilst chromosomes XIII and III were more highly conserved. As the variation graph can be continuously evolved by incorporating variants from additional strains, changes to the N:SL ratio across each chromosome graph can be monitored to study how the degree of sequence diversity changes with the addition of each strain.

The performance of mapping the NCYC1006 sequence reads against the NCYC variation graph was determined solely by the distribution of the alignment scores. Alignment scores are widely used when conducting pair-wise and multiple sequence alignments to denote the level of sequence similarity amongst the sequences analysed. Alignment scores account for the length of the sequence and the level of sequence similarity by penalising any mismatches and gaps in alignment. Consequently, as bases that match are rewarded one point, the maximum alignment score for a perfect alignment is the length of the sequence itself. For the purpose of read mapping, as sequence reads may be aligned against many different regions of the reference genome, the highest scoring alignment is selected by the mapping algorithm as the optimal primary alignment for a read. The use of alignment scores to determine mapping provides confidence that the read is aligned to its true location. The *vg* mapping algorithm was found to perform relatively well at aligning the NCYC1006 sequence reads against the haploid variation graph, a particularly strong result when taking into consideration that NCYC1006 is a tetraploid strain and therefore poses the scope for a wider range of sequence diversity. The majority of reads had alignment scores greater than 100 whilst the maximum score was 118 which was indicative of good alignment quality in mapping.

The Bergstrom strains were selected as the initial strain set for the construction of a pan-genome variation graph reference structure for *S. cerevisiae* as these strains have been extensively characterised and were derived from various divergent sub-populations thus likely to contain a broad pool of variants. The raw read coverage across the strains was found to be multifarious which strongly indicated that the DNA libraries had not been normalised during sequencing preparation. DNA library normalisation is a critical step that allows for uneven concentration of DNA libraries to be equalised, particularly

when multiplexing, as high DNA library concentrations can lead to excessive read depth. For example, this phenomenon was seen with UWOPS83-787.3 and UWOPS87-2421 which resulted in over 100 times more sequence reads than for the other strains. On the other hand, low DNA library concentrations result in low read depth, as seen with YJM981, which can have a marked impact on the accuracy of read mapping, variant calling and any further downstream analyses.

The efficacy of read mapping was primarily determined by two factors, the number of sequence reads that were able to align against each reference and the alignment scores. The use of the *vg* Bergstrom variation graph and the reference graph was found to far outperform the linear reference genome as reference structures in terms of the number of reads that were able to align. The stark difference in mapping seen between the reference graph and the linear reference genome, as both references only represent a single haplotype at each locus, indicated that the *vg* mapping algorithm has greater sensitivity than that used by *BWA*. The Bergstrom variation graph was found to align the greatest proportion of reads suggesting that the prior inclusion of variants from the Bergstrom strain population reduced bias in mapping enabling more reads to align against the reference. These findings concurred with the study on *vg* which compared mapping for 100,00 paired reads simulated from the mosaic, soil strain, SK1, against the *vg* graph-based references, both with and without variants pertaining to SK1 and the linear reference genome. This study showed that the variation graph had the greatest alignment sensitivity and that there was little difference in mapping against the variation graph and the graph without the SK1 variants (Garrison *et al.,* 2018). The GAGH4 pilot study (Novak *et al.,* 2017) comparing various genome graphs also reported that variation graphs outperformed the linear reference genome with a greater number of perfectly mapped reads against the human genome. Read mapping across all three reference structures overall was found to be inadequate as vast amounts of the sequence reads were completely unable to align. One reason for such observation is poor sequence read quality, therefore greater quality control was carried out in the later experiments. The total number of reads processed by *BWA* was found to be much greater than both the reads output from *vg* and found within the FASTQ files. Further investigation showed that the additional *BWA* reads were due to the presence of chimeric alignments. Chimeric reads align to two distinct portions of the reference genome without any overlaps and may be indicative of structural variation. As the

vg mapper does not produce chimeric alignments, these were removed by filtering the BAM file prior to analysis to allow for fairer comparison between the different mapping software.

The variation graph was also found to have the best alignment quality of the three references when judged based on the alignment scores. This supported the findings from a previous study (Jones, 2016) which also compared the alignment scores between the linear reference genome and the *vg* graph-based references for the Bergstrom strains and found that the variation graph had the highest alignment scores across all 19 strains. Initially, this was thought to be due to the inclusion of variant alleles, allowing for a greater number of reads to align to their true location, though further experiments subsequently showed that there were hidden differences in the scoring parameters that bias the alignment scores in favour of *vg*. Thus, accurate conclusions could not be drawn from this experiment when comparing the alignment quality from *vg* against *BWA* but this will be further explored in-depth in the following chapter. However, comparison of the alignment scores between the reference graph and the variation graph, both of which utilise the same default scoring parameters in *vg*, showed that reads aligned with greater alignment scores against the variation graph. This could be due to reads aligning to the same position against both graphs but with the presence of variant alleles in the variation graph allowing for more base matches. Such a scenario would result in improved alignment scores, or alternatively variant alleles could decrease ambiguity in mapping, allowing reads that were aligned at a different position in the reference graph to align to their true location in the variation graph.

Visualisation of the NCYC and Bergstrom variation graph highlighted the differences in the structure of both graphs caused by the presence of population-specific variants. This further re-iterated the importance of selecting as many diverse strains as possible when constructing the variation graph in order to increase variant representation within the graph which in turn can further improve the accuracy of mapping and eliminate reference allele bias. Also, the use of similar strains introduces a lot of data redundancy in an already computationally expensive process. Given more time, it would have been beneficial to explore optimal methods for selecting strain subsets when constructing a pan-genome variation graph, perhaps by using phylogenetic analysis to identify strains with greater sequence diversity. However, due to the time constraints of the project,

this work could not be carried out.

The mitochondrial genome was of particular interest as the N:SL ratios of the NCYC variation graph indicated that it possessed a greater degree of variation than the nuclear chromosomes. The mitochondrial DNA in *S. cerevisiae* is of varying size, at its minimum approximately 75 Kb, and therefore contains at least 15% of the total DNA content of the genome (Williamson, 2002). Despite this, only 2-5% of the sequence reads aligned to the mitochondrial genome, suggesting that many mitochondrial reads remain unaligned, likely due to a high degree of intragenic sequence diversity. These findings are in concordance with previously published studies on the *S. cerevisiae* mitochondrial genome which found extensive genome content variation, particularly due to horizontal gene transfer of mobile elements between populations (Goddard *et al.*, 1999) and the accumulation of intergenic sequences (Groth *et al.*, 2000). In addition, the inheritance of the mitochondrial genome is bi-parental (Solieri, 2010) and the rate of mutation is higher than that of the nuclear genome (Foury *et al.*, 2004). The combination of the above mentioned factors contribute to the vast genetic diversity of the mitochondrial genome (Wolters *et al.*, 2015).

A greater proportion of reads from the NCYC strains was found to align against all three references, including the Bergstrom variation graph, than the reads from Bergstrom strains themselves. This observation suggested that the sequence read quality of the Bergstrom strains was lower than that of the NCYC datasets. Quality control analysis revealed that many Bergstrom reads were of low sequence quality with the presence of adapter contamination. Consequently, the Bergstrom sequence reads had to be trimmed whilst the NCYC strains had consistently high quality sequences resulting in a greater proportion of reads that are able to align while unprocessed. This also explained the disparity in mapping seen between Y12 and NCYC3630, the latter a derivative of Y12, in which there was a 24% difference in the proportion of reads that aligned against the graph-based references even though both strains should have had very similar genome sequences. The high sequence quality of the NCYC datasets also enabled reads to align with greater alignment scores for all three reference structures. However, as the maximum alignment score is determined by the length of the sequence, the consistently higher scores in *vg* were indicative of systemic bias in the *vg* scoring parameters.

Read trimming was found to vastly improve the sequence quality of the Bergstrom strains, albeit resulting in a vast number of the sequence reads being discarded, which enabled a greater proportion of reads to align. Trimming both read sets also increased the relative proportion of Bergstrom reads that were able to align when compared to the NCYC strain reads. The differences in the proportion of reads that were able to align against each reference was also reduced for the trimmed reads, reinforcing the idea that sequence read quality plays a significant role in mapping. For the NCYC strains, the greater numbers of mapped raw reads than trimmed reads were hypothesised to be likely due to a combination of trimming removing the lower confidence reads that would otherwise be able to align and discarding one half of a paired read preventing the other pair from mapping. This was confirmed when aligning the sequence reads discarded during trimming for NCYC78 and NCYC97 as only half of the reads would still have been able to align. Thus, the removal of these reads through trimming increased the proportion of reads that mapped. These findings indicated that the choice of whether or not to carry out read trimming before variant graph mapping should be determined after running quality control analysis on the reads, as trimming of high quality reads may result in incomplete read mapping.

Both *vg* and *BWA* utilize similar scoring systems. However, many crucial differences were identified in their default scoring parameters and penalties that are not explicitly stated in the algorithm descriptions, but which bias the scores when comparing the read alignment quality between the two mappers. These differences identified in the scoring parameters demonstrate that *BWA* employs more stringent penalties in scoring alignments in comparison to *vg* which applies unnecessary bonuses to its alignment scores. There were also discrepancies between the alignments produced by *BWA* and the subsequent alignment scores, as the scores were found to reflect the local alignment from the Smith-Waterman extension phase instead of the final alignment score from the banded global alignment that is used to produce the CIGAR string in the output BAM file. The local alignment score is used for computation of the mapping quality scores and may differ from the global alignment score if the Maximal Exact Match (MEM) used during the Smith-Waterman extension phase produces a sub-optimal alignment (Li, 2013). *BWA* does not output the final global alignment score, as does *vg*, making it difficult to compare the mappers as the *BWA* alignment score may not reflect the final alignment for certain reads. These findings establish alignment scores as biased

and inaccurate and therefore not a viable measure for comparison of the quality of read alignments across various mapping software. Instead, a matched CIGAR string was used as the basis of a method for measuring the alignment quality from different software, as discussed in the following chapter.

The simulation study exploring various factors that impact read mapping identified that the length of the sequence read, the type of variant and the position of the variant all impact on the ability of a sequence read to align against a reference sequence. Alignments for the simulation study were carried out in *vg* to obtain a better understanding of the graph-based mapping algorithm. However, it would have been useful to extend the comparison by aligning the simulated reads in *BWA* in order to identify the differences in the extent to which each of the factors impact mapping across the two software. Mutations were simulated on the sequence reads after generating the reference sequence which resulted in a greater number of read specific mutations. The different mutation types could have been simulated on the reference genome prior to the generation of sequence reads to mitigate this issue. The use of this alternative approach could have also allowed for the study of the impact of sequencing errors on read mapping. Mutations at either ends of a sequence read sequence did not have an impact on mapping as they were treated as sequencing artefacts by the *vg* mapping algorithm and therefore masked out through soft-clipping during alignment. Soft clipping refers to the masking of bases that are not aligned against a reference sequence. Point mutations had the greatest impact on read mapping reaffirming the inability of sequence reads to align against the highly polymorphic regions of the genome. Point mutations within the sequence reads mimicked reads from strains with a greater degree of sequence diversity, therefore, the more divergent the strain, the less likely it is able to align against the reference genome. Read length was also identified to be a major determining factor as increasing the read length appears to increase tolerance to variants within the sequence read. Therefore, the use of long reads may be a useful approach to improve mapping. However, a study on the *Seven Bridges Graph Genome* toolkit (Rakocevic *et al.*, 2019) found that both short reads and PacBio long reads were completely unable to align against regions spanning large-scale structural variants in a linear reference genome whereas using a graph genome with the population variation information incorporated enabled the alignment of both datasets. Against the linear reference, the reads were also found to misalign across the structural variant breakpoints, due to the imperfect sequence microhomology

within these regions leading to incorrect genotyping during variant calling (Rakocevic *et al.*, 2019). The results seen in this study further support the benefits of using genome graphs as reference structures to improve mapping and variant calling.

2.5 Conclusion

The use of variation graphs as reference structures improved both the quantity of sequence reads that were able to align to them and the quality of the alignments, improving the accuracy of read mapping which will also likely enable greater accuracy in variant calling. These findings were consistent with the previous work carried out on genome graphs and the *vg* software, which showed that the use of variation graphs can overcome several of the limitations faced by linear mapping methods (Novak *et al.*, 2017, Garrison *et al.*, 2018; Jones, 2016). The significant increase in sequence read mapping from the linear reference to the reference graph also indicated that the *vg* mapping algorithm has greater sensitivity in mapping than *BWA* and that sequence reads are able to align more easily against a graph-based reference structure than the conventional linear reference, the most popular method for read mapping. It is currently unclear as to what aspect of the mapping algorithm allows it to exhibit a greater preference towards aligning reads against a graph-based reference structure even when there are no variants present. The proportion of mapped reads increased, as expected, when alignments were made against the variation graph, which includes known variants within the species population. The inclusion of variants was also demonstrated to positively impact the alignment scores as the variation graph had a higher number of reads that mapped with greater alignment scores in comparison to the reference graph, suggesting the decreased ambiguity in mapping allowed for more reads to align to their true location. The use of alignment scores provided a biased measure in the comparison of alignment quality and therefore, could not be used to compare alignments between *vg* and *BWA*. Instead, sequence identity scores can be used to obtain an accurate depiction of alignment quality across various graph-based and linear mapping algorithms, which will be explored further in the following chapter.

There were several different factors identified to impact the ability of sequence reads to align against a reference. Poor sequence read quality was found to be a major limitation to mapping, which could be improved by trimming the raw reads. However,

this may still leave large portions of the reference relatively unaligned. The size of the read sequences, the size of the variants and the type of variants also played a key role in read mapping. The presence of several SNPs in close proximity was found to prevent mapping which could be overcome through the use of variation graphs by accounting for known variant alleles which reduces reference bias. These findings further advocate for the use of variation graphs as the primary reference structure.

Chapter 3

Comparison of the Performance of Read Mapping and Variant Calling Algorithms Utilised by Graph Genome Software

3.1 Introduction

The primary focus for graph genome software is to be utilised in large-scale population level sequencing studies to genotype and characterise variants in the human genome. The studies published on *vg* (Garrison *et al.*, 2018), *Seven Bridges Graph Genome* toolkit (Rakocevic *et al.*, 2019), *Graphtyper* (Eggertsson *et al.*, 2017) and *BayesTyper* (Sibbesen *et al.*, 2018) have shown that the advantages of using variant aware graph-based reference structures far outweighs the linear reference genome. This strongly favours the adoption of variation graphs as the standard practice for variant discovery across all genomes, particularly yeast genomes which display a high level of both inter- and intragenic variation. These four graph genome software were chosen as the variant calling performance of the *Graph Genome* toolkit has been compared with *Graphtyper* and *BayesTyper* (Rakocevic *et al.*, 2019) and the *vg* toolkit has also outlined methods for integration with *Graphtyper* (Garrison *et al.*, 2018). However, there has yet to be a study that compares the accuracy of variant prediction across all four graph genome software. Therefore, it is crucial to carry out a comprehensive analysis that tests the relative performances of read mapping and variant calling in order to identify the most optimal software for the analysis of yeast genomes. An overview of each graph genome

software has been depicted in Table 3.1.

	vg toolkit	Graph Genome toolkit	Graphtyper	BayesTyper
Graph can be visualised	✓	✗	✗	✗
Alignments can be accessed	✓	✓	✗	✗
Linear alignment not required	✓	✓	✗	✗
Works with whole genome	✗	✓	✗	✓
Compatible with non-human genomes	✓	✓	?	?
Discovers novel variants	✓	✓	✓	✗

Table 3.1: **Overview of Graph Genome Software Capabilities.** Comparison of the main differences and limitations of four key graph genome software.

The aim of this chapter is to carry out a detailed analysis of the performance of the read mapping and variant calling algorithms employed by *vg*, *Graph Genome Pipeline*, *Graphtyper* and *BayesTyper* against the conventional linear pipeline, which utilises *BWA* for mapping and *FreeBayes* for variant calling. The *FAT-CIGAR* toolkit was developed to obtain the exact surjected and non-surjected alignments for both linear and graph-based mapping software. The various mapping algorithms were compared in terms of the percentage of mapped reads achieved and sequence identity scores were used to determine the quality of read alignments. A large-scale simulation study was carried out to assess the accuracy of variants calls from each software against the truth-set for 1,000 datasets generated using the *TreetoReads* software, each containing 19 simulated genomes that mimicked the Bergstrom strains analysed in Chapter 2. The number of true positive, false positive and false negative variants were compared to determine the level of precision and recall for each software.

3.2 Methods

3.2.1 The *FAT-CIGAR* Software

The CIGAR string information found within BAM files is a succinct representation of a sequence reads alignment against a reference (see Figure 3.1) and therefore, can provide an unbiased metric for comparison of the accuracy of alignments across the various mapping software. Sequence read alignments are characterised by the length of the sequence succeeded by one of the following strings, each describing an associated operation: 'M' (base match), 'I' (insertion), 'D' (deletion), 'S' (soft-clipped base) and 'H' (hard-clipped base). Hard-clipped bases are masked bases that are removed from the sequence read during alignment (Li *et al.,* 2009). However, the CIGAR string does not distinguish between matched and mismatched bases hence masks any mismatched base that aligns against the reference sequence within a run of 'matched' bases. In order to carry out direct comparison of alignments generated by *vg*, the *Graph Genome* toolkit and *BWA*, it was crucial to have an exact CIGAR string representation of alignments that distinguishes between matched and mismatched bases. The *FAT-CIGAR* Python toolkit (https://github.com/prithikasritharan/FAT-CIGAR) was developed to re-construct the exact CIGAR string, also referred to as the FAT-CIGAR string, utilizing the alignment information within the BAM file for both linear and variation graph references.

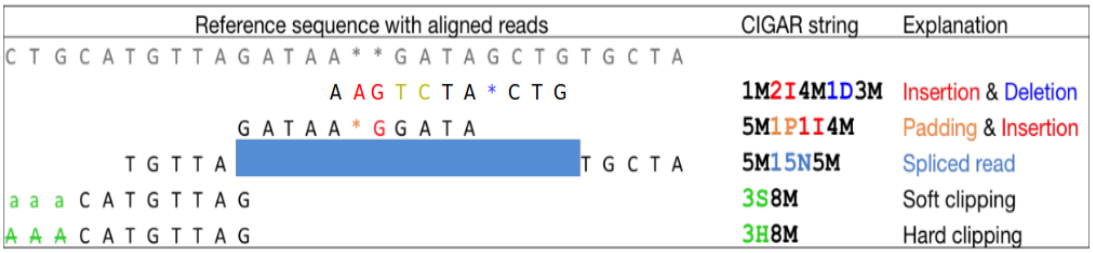


Figure 3.1: **CIGAR String.** This figure demonstrates how the alignment of sequence reads against a reference sequence can be represented using the CIGAR string. The associated operations for each alignment have also been specified next to the CIGAR string. This figure was adapted from Dündar *et al.*, 2015.

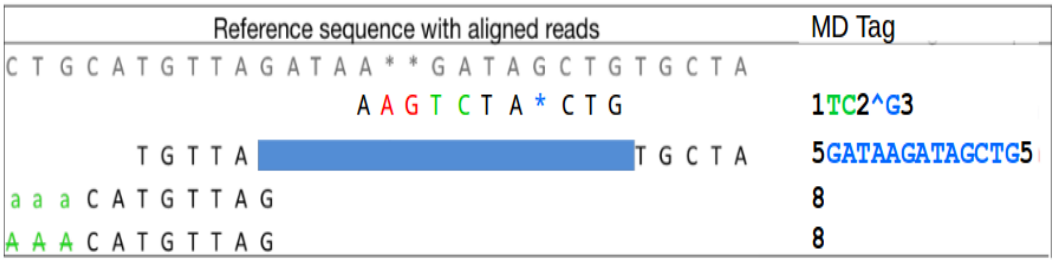


Figure 3.2: **MD Tag.** This figure shows an example of the alignment of sequence reads against a reference sequence followed by the MD tag representation of the alignment. This figure was adapted from Dündar *et al.*, 2015.

FAT-CIGAR Toolkit: Linear Alignments

BAM files generated via *BWA* contain an MD tag, which is similar to the CIGAR string but provides information regarding mismatched bases and deletions within the alignment by replacing it with the reference base. The MD tag (see Figure 3.2) represents matches solely by their sequence length, mismatches by the mismatched base within the sequence read and deletions by a '^' followed by the deleted base in the reference genome. The MD tag cannot be used directly for comparison, however, as its primary purpose is to call SNPs without the need for a reference, therefore, it only takes into consideration the bases that align against the reference. This results in the masking of any insertion sequences within the alignment and the position of the following matches are shifted downstream. The *FAT-CIGAR* toolkit combines the MD tag and the CIGAR string information to construct the FAT-CIGAR string for alignments. Not all reads produce an MD tag in their BAM files. In such cases, the *SAMtools calmd* function can be utilised to generate the MD tag prior to running the *FAT-CIGAR* toolkit.

For alignments against a linear reference genome, the *FAT-CIGAR* toolkit must be run with the *linear* command followed by the input and output BAM files which are required arguments. The input BAM file is initially checked to ensure the file is indexed. If not, the file is sorted and an index is generated using the pysam module in Python (Li *et al.*, 2009). The CIGAR string and MD tag for each read is extracted and the MD tag is transformed into the FAT-CIGAR string as follows:

- an 'M' is added to the end of numbers that represent matched bases.
- mismatched bases are replaced with the length of mismatch plus 'X' instead of

the reference bases.

- deletions are replaced with the length of deletion plus 'D' instead of the deletion sequence.
- any soft and hard clipped bases are added to both ends of the read.

The CIGAR string is used to identify the exact positions of the insertion sequences and to insert them within the modified MD tag. Then the number of matches on either side of the insertion sequence is recalculated to produce the FAT-CIGAR string (see Figure 3.3). The alignments are written to the output BAM file with the FAT-CIGAR string, overwriting the CIGAR string information as default. The *-xg* option can be specified by the user in order to preserve both the CIGAR and FAT-CIGAR strings in the output BAM file, by assigning the FAT-CIGAR string to the XG tag.

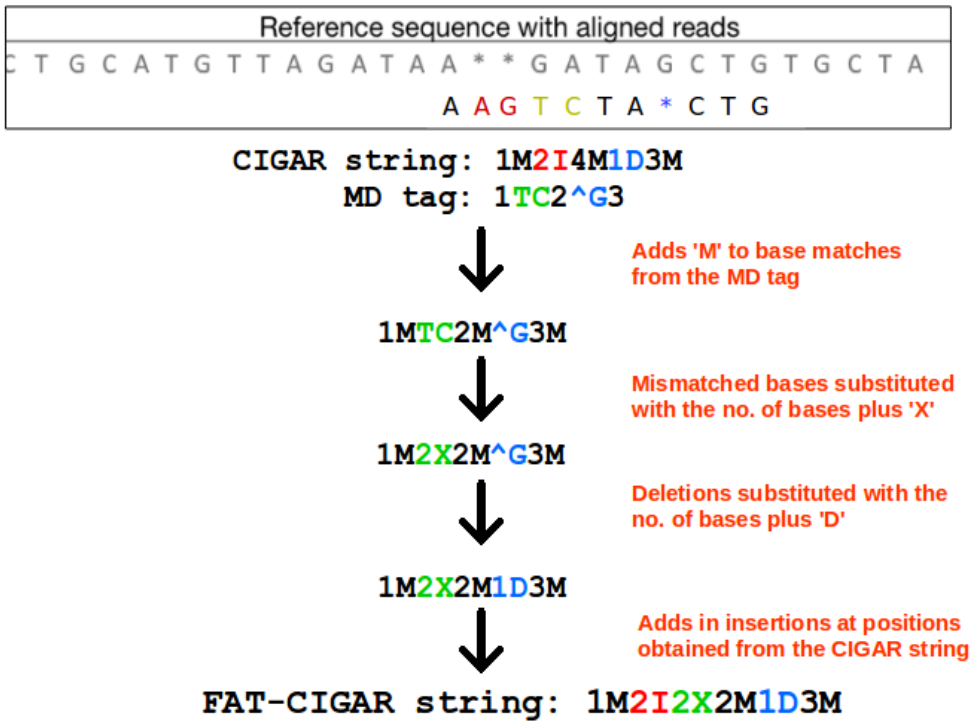


Figure 3.3: **Constructing the FAT-CIGAR String from a BAM file.** This figure shows an example of how the *FAT-CIGAR* toolkit utilises the CIGAR string and MD tag information from within the BAM file to construct the FAT-CIGAR string representation of the alignment in which each operation is clearly distinguished. This figure was adapted from Dündar *et al.*, 2015.

FAT-CIGAR Toolkit: *vg*

The *vg surject* function is used to convert GAM files into BAM file format. However, the CIGAR string within the resulting *vg* BAM file depicts the surjected alignment against the S288c linear reference genome rather than the alignment path through the variation graph. The *vg* toolkit enables GAM files to be converted into JSON format, which contains detailed information on the individual path within the graph that each sequence read is mapped to. Therefore, exact read alignments can be reconstructed based on the *from_ length* (the length of the sequence within the variation graph) and *to_ length* (the length of the sequence within the read) information in the mapping edits array of the JSON file (see Figure 3.4). Matched bases have equal *from_ length* and *to_ length* and no sequence whilst mismatches have equal *from_ length* and *to_ length* with a sequence provided. Insertions and deletions have different *from_ length* and *to_ length* values with the inserted sequence specified for insertions. Sequences are not hard-clipped in *vg* but soft-clipped bases resemble insertions at the start or end of sequence reads. The *FAT-CIGAR* toolkit utilises the above information in the JSON file path edits to reconstruct the non-surjected FAT-CIGAR string for alignments against the *vg* variation graph. The JSON file needs to be obtained prior to running *FAT-CIGAR*, using the *vg view* function on the GAM file with the *-aj* option.

To obtain the FAT-CIGAR strings for *vg* alignments, the *graph_ vg* command should be utilised. The *FAT-CIGAR* toolkit takes the JSON file, the surjected BAM file and an output BAM file as required arguments. The BAM file is checked to ensure it is indexed. If not, the BAM file is sorted and an index is generated. For each read within the BAM file, the JSON file is required to be searched for the pertaining alignment information. As the mapping information is contained within large, nested and unsorted arrays, a hash table was implemented to index the JSON file to allow for more efficient searching. The hash table is built with the default size of 2**24 to ensure that it is large enough to store the alignments and the read names are computed as the hash string. The hashed read name is divided by the table size and the resulting modulus is used to determine the index position within the hash table. The position of the read data array in the JSON file is stored as the key. If a collision (more than one key at an index within the hash table) occurs, collisions are resolved by storing the keys at that index in a linked list through chaining. In order to optimise memory space, the first entry is never wrapped in a node. Once the hash table is built, the BAM file is read in sequence

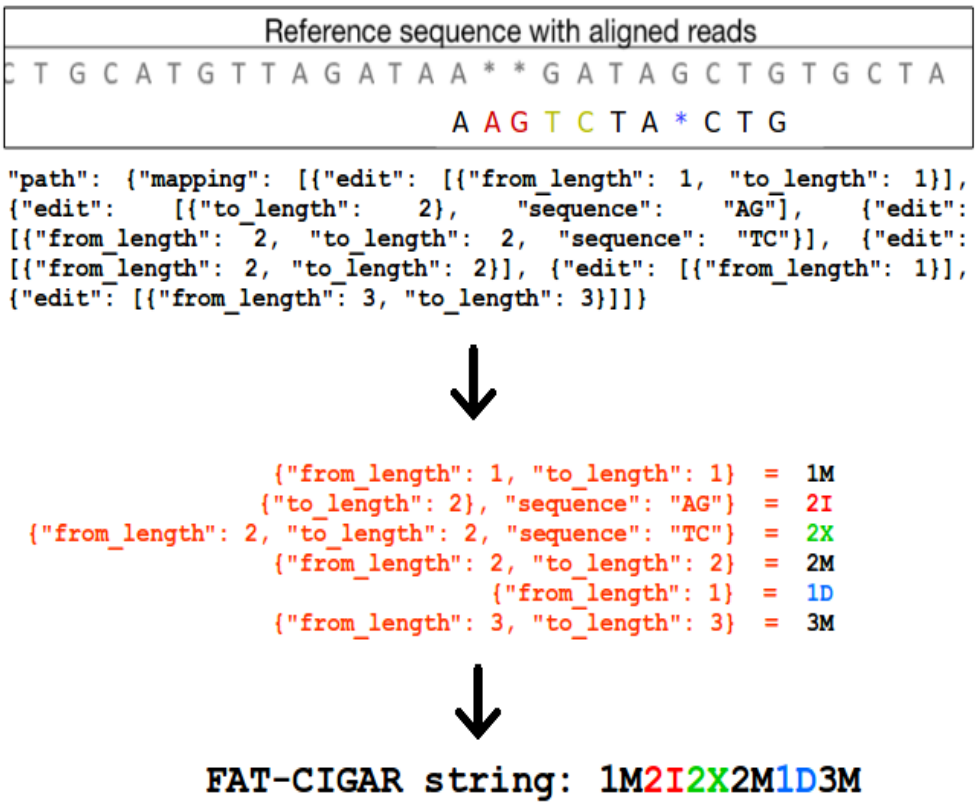


Figure 3.4: **Constructing the FAT-CIGAR String for a *vg* Alignment.** This figure is a simplified example of how the *FAT-CIGAR* toolkit utilises the mapping path information from within the JSON file to construct the FAT-CIGAR string. It highlights how the associated operation in the FAT-CIGAR string is obtained from each path edit information. This figure was adapted from Dündar *et al.*, 2015.

read by sequence read and any sequence reads mapped to the reverse strand are reverse complemented to obtain the original sequence.

A search function is implemented to search the hash table for the corresponding JSON object for the read in order to construct the FAT-CIGAR string. The search function computes the hash from the read name for each read and looks it up in the hash table. If the key is found (i.e. there is a matching JSON object for that read), the JSON data at that position is read from the file and the read name and sequence are compared to verify the match. If there are multiple positions, each entry in the linked list is checked until a match is found. The JSON data at that position is passed to a function which uses the *to_length* and *from_length* information and constructs the FAT-CIGAR string for the alignment as described above. If the read is mapped to the reverse strand, the FAT-CIGAR string is also reversed to follow the BAM file protocol.

As the *vg* BAM files do contain the alignment score, the alignment score is also parsed out of the JSON object and set as the 'AS' tag in the output BAM file. If a read is unmapped and therefore has no mapping edit information in the JSON file, a "*" is returned as the CIGAR string and the AS tag is set to 0. As default, the reads will be written to the output BAM file with the non-surjected FAT-CIGAR string. The *-xg* option can be specified by the user to write the non-surjected FAT-CIGAR string to the XG tag. This option also overwrites the surjected CIGAR string with the non-surjected CIGAR string. In order to do this, the *-c/--xg_to_cigar* function converts the FAT-CIGAR string into a CIGAR string by splitting the string to merge any matches "=" and mismatches "X" into an "M". The length of the matched sequence is recalculated before joining the string together to form the non-surjected CIGAR string.

***FAT-CIGAR* Toolkit: *Graph Genome* Toolkit**

The BAM file output from the *Seven Bridges Graph Genome* toolkit also represents read alignment information as a surjected CIGAR string and the XG tag contains the non-surjected FAT-CIGAR string. It was therefore necessary to obtain non-surjected CIGAR string alignment to enable comparison to other read mappers. In addition, the XG tag was absent for any reads that aligned as perfect matches against the variation graph, hence the XG tag had to be generated for these reads to obtain the non-surjected FAT-CIGAR string.

The *FAT-CIGAR* toolkit is run with the *graph_sb* command for alignments from the *Graph Genome* toolkit, with the input and output BAM files as required arguments. For each read, the *FAT-CIGAR* toolkit checks whether the XG tag is present and if not, the "M" in the CIGAR string is substituted with an "=" and set as the XG tag generating the non-surjected FAT-CIGAR string. The same function as described previously for the *vg* alignments is utilised to construct the non-surjected CIGAR string from the XG tag. The original surjected CIGAR string is overwritten with the non-surjected CIGAR string in the output BAM file.

FAT-CIGAR Toolkit: Extended Functionality

In order to increase the usability of the *FAT-CIGAR* toolkit and make it more widely applicable, the following two optional functions were also added for linear alignments: the *-g/--global_score* to calculate the global alignment score and the *-cs/--cs_tag* function to generate the CS tag. As noted in Section 2.4, the alignment scores output by *BWA*, the most commonly used read mapper, are calculated from the local alignment during the Smith-Waterman extension phase instead of the final banded global alignment. Consequently, the scores for many of the reads did not accurately reflect the actual alignment. If specified by the user, the global alignment score is calculated from the FAT-CIGAR string using the same scoring system employed by *BWA* as outlined previously in Chapter 2. The CS tag (see Figure 3.5) is an improved representation of the MD tag to allow for easier parsability and is output by the Minimap2 software (Li, 2018). This option was added to the *FAT-CIGAR* toolkit to further improve inter-compatibility with output from other read mappers. The FAT-CIGAR string was converted into the CS tag as follows: matched bases are represented as ":" plus the length of the match, mismatches are represented as "*" followed by the mismatched base sequence, insertions are represented as "+" followed by the inserted bases and deletions are represented as "-" followed by the deleted bases.

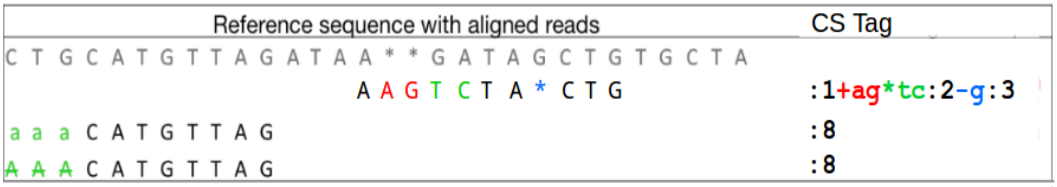


Figure 3.5: **CS Tag.** This figure shows an example of the alignment of sequence reads against a reference sequence followed by the CS tag representation of the alignment. This figure was adapted from Dündar *et al.*, 2015.

3.2.2 Evaluating the Performance of Mappers From Various Graph Genome software

The development of the *FAT-CIGAR* toolkit allowed for an unbiased measure of the accuracy of read alignments between various different mappers. Of the four graph genome software, only *vg* and the *Graph Genome* toolkit were able to provide read

alignment information, as noted in Table 3.1. Thus, the Bergstrom *S. cerevisiae* strains (Bergstrom *et al.*, 2014) discussed in Chapter 2 were used to evaluate the accuracies of read alignments from both graph genome software against alignments from *BWA*. The FAT-CIGAR string was used to compare the differences in alignment quality against the reference graph, variation graph and linear reference genome.

Seven Bridges Graph Genome Toolkit

The *Seven Bridges Graph Genome* toolkit was downloaded from the Seven Bridges website (https://www.sevenbridges.com/graph-genome-academic-release/). The downloaded tar binaries are wrapped in Docker containers and distributed as two compressed Docker images: the *bpa* aligner v0.9.1.1 and the *rasm* variant caller v0.5.20. The Docker images required a machine with a Linux operating system and a microprocessor capable of handling Advanced Vector Extensions 2 (AVX2) instruction set architecture in order to run. The tar files were loaded into Docker and accessed by running their Docker image IDs.

The S288c reference graph was initially created by running the *bpa* aligner with just the S288c reference genome FASTA file and the FASTQ files containing the trimmed paired-end sequence reads for each of the 19 Bergstrom strains (see Figure 3.6 for a summary of the methods). As the *Graph Genome* toolkit creates each graph in memory before alignment, the graph was re-created before mapping for all 19 strains. A few sequence reads were removed from both the forward and reverse FASTQ files as a memory bug in the software caused the aligner to fail when these reads were present in the following five strains: W303 (IL29_4505:7:92:17915:7957#8), YPS128 (IL29_4505:7:16:3889:7672#1), DBVPG6765 (IL29_4505:7:33:16078:9366#5; IL29_4505:7:39:9332:6247#5; IL29_4505:8:3:10070:14525#5), DBVPG1788 (IL29_4505:8:55:8359 :17891#4) and DBVPG1106 (IL29_4505:2:20:3032:15548#1). The resulting BAM files containing the aligned sequence reads for each strain were sorted and indexed using *SAMtools* (Li *et al.*, 2009) *sort* and *index* functions. The *rasm* reassembly variant caller was run with the sorted BAM file and the S288c reference FASTA file to call variants for each strain. The output VCF files were decomposed using both the *decompose* and *decompose_blocksub* functions to break down the multi-allelic variants into constituent SNPs and indels and normalized using the *Vt* software. The normal-

ized VCF files were validated using *GATK ValidateVariants* and the *CombineVariants* function was used to combine the variants from all 19 strains into a single multi-sample VCF file.

In order to create the Bergstrom variation graph, the *bpa* aligner was run with the S288c reference FASTA file, the merged VCF file containing all the variants within the Bergstrom strain population and the paired-end FASTQ files for each strain. Sequence reads also had to be removed from both FASTQ files for the following three strains: UWOPS87-2421 (HS3_6632:2:2105:13226:155086#2; HS3_6632:2:2108:1521:133487#2), UWOPS03-461.4 (IL29_4505 :2:101:19612:9286#6) and UWOPS83-787.3 (HS3_6632:2:2101:10238:137772#1; HS3_6632:2:1206:4335:176399#1; HS3_-6632:2:1107:15487:137370#1). The output BAM files containing alignments against the variation graph were sorted and indexed as before. For each strain, the *FAT-CIGAR* toolkit was run on BAM files containing alignments from the reference graph and variation graph to obtain the non-surjected CIGAR and FAT-CIGAR strings and the run time of the toolkit was monitored. The variant caller was run with the sorted BAM file, S288c reference FASTA file and the multi-sample VCF file used to construct the Bergstrom variation graph in order to call variants from alignments against the variation graph itself. The VCF files for each strain were decomposed and normalized using *Vt* and validated with *GATK ValidateVariants* to be used to compare the accuracy of variant calling against other graph genome software.

Variation Graph (*vg*) Toolkit

The S288c reference graph and Bergstrom variation graph were also recreated in *vg* v1.18 in order to update the software to its latest version due to modifications to the core algorithm, such as the use of graph BWT (gBWT) for graph construction instead of the graph extension of positional BWT (gPBWT) in earlier versions.

The reference graph was created using the S288c reference FASTA file with *vg construct*, limiting graph construction to each individual chromosome (see Figure 3.6 for a summary of the methods). The nodes across the 17 per-chromosome graphs were joined together with *vg ids*. Both the XG and GCSA2 indexes were generated for each

graph without pruning nodes due to the absence of variants in the graph. The trimmed sequence reads for all 19 strains were mapped against the S288c reference graph with *vg map* and the read alignments were written out to GAM files. The GAM files were sorted and indexed with *vg gamsort* then converted into JSON format to obtain the alignment information. The *vg surject* function was also used to obtain BAM files, which were sorted and indexed. Variant calling was carried out by initially breaking up the sorted GAM files using *vg chunk* into per-chromosome chunk files. Any ambiguous or secondary read mappings were filtered out of the GAM files with *vg filter* using the default parameters suggested by *vg* (*-r 0.90 -fu -m 1 -q 15 -D 999*) to improve the precision of variant calling. The reads within the filtered GAM files were embedded as paths onto the graph to create augmented variation graphs using *vg augment* with the following parameters: *-a pileup* (computes read pileup), *-S* (read support file for variants) and *-Z* (contains graph translation between base graph and augmented graph). Variant calling was carried out on the augmented graphs with *vg call* using the base graph (*-b*), the read support file (*-s*) and the translation file (*-z*) to produce per-chromosome VCF files for all 19 strains. Each VCF file was decomposed and normalized using *Vt* and sorted using the *vcf-sort* function from VCFtools v0.1.16 (Danecek *et al.*, 2011). The sorted VCF files were concatenated into single strain VCF files using *CatVariants* from *GATK*, validated using *ValidateVariants* and merged with *CombineVariants* to produce a multi-sample VCF file.

The multi-sample VCF file containing the variants from the Bergstrom strain population was used to construct the Bergstrom variation graph along with the S288c reference FASTA file. The node id spaces were joined across the per-chromosome graphs and the XG index was generated. In order to reduce the complexity of the graph, the *vg prune* function was used to remove any edges which induced more than three bifurcations within a 24bp sequence before the GCSA2 index was generated. The trimmed sequence reads for all the strains, except UWOPS83-787.3 and UWOPS87-2421, were aligned against the variation graph and variants were called from the alignments using the same method as previously described for the S288c reference graph. Due to the large file size of the FASTQ files for UWOPS83-787.3 (76,271,936 reads, forward reads: 7.2G & reverse reads: 7.1G) and UWOPS87-2421 (99,868,770 reads, forward reads: 9.2G & reverse reads: 8.9G), the *vg* mapper would terminate producing a run time error as a result of insufficient memory. Therefore, the FASTQ files for both strains were split into

multiple smaller files each containing around 500,000 reads. The split FASTQ files were aligned against the variation graph and the resulting GAM files were concatenated into a single GAM file before sorting and variant calling. The *FAT-CIGAR* toolkit was run with the surjected BAM files and JSON file as input to obtain the non-surjected CIGAR and FAT-CIGAR strings for alignments against the reference graph and variation graph. The run-time of the *FAT-CIGAR* toolkit was also monitored for each strain.

Comparison of Alignments From *BWA*, *vg* & the *Graph Genome* toolkit

In order to compare the ability of each read mapper to align sequence reads to a reference structure, the percentages of mapped reads against the S288c reference and Bergstrom variation graphs created in both *vg* and the *Graph Genome* toolkit and the linear reference genome were obtained from the resulting five sets of BAM files using *SAMtools*. The *R* v3.6.0 software (R Core Team, 2017) was utilised to create a line graph showing the percentage of mapped reads across the different reference structures for all 19 strains.

In order to identify whether aligning against a variation graph can reduce ambiguity in mapping, resulting in an increased number of true, perfectly mapped reads (i.e. reads that align against the reference as an exact match), sequence identity scores were calculated for alignments against all five reference structures. Sequence identity scores can be calculated from the CIGAR string to provide an indicator of sequence similarity between the read and the reference. However, the masking of mismatches within the CIGAR string biases the scores, over-inflating the number of true, perfect mappings. Sequence identity scores were calculated from both the CIGAR and FAT-CIGAR string to evaluate the alignments produced from each mapper. Initially, *R* was used to calculate the sequence identity scores however, regular expression processing across several large data frames in *R* was extremely slow. Therefore, a custom Python script (https://github.com/prithikasritharan/SeqIdCalc) was written to calculate the scores using the BAM files. The sequence identity score, Sc, for the CIGAR string was calculated as follows:

$$Sc = \frac{\text{no. of matches}}{\text{no. of matches} + \text{no. of insertions} + \text{no. of soft-clips}} \tag{3.1}$$

and set as the ZA custom tag within the BAM file. Any hard-clips and deletion sequences are ignored in the calculation as these bases are removed from the sequence

read. The scoring for the FAT-CIGAR string, Sf, was calculated as follows:

$$Sf = \frac{\text{no. of matches}}{\text{no. of matches} + \text{no. of mismatches} + \text{no. of insertions} + \text{no. of deletions}} \tag{3.2}$$

and set as the ZB custom tag. The BAM files were loaded into R and converted into data frames to create histograms of the sequence identity scores.

The different data frames were merged in R in order to obtain common reads that mapped across all five references. In order to identify the similarities in mapping across all the references, the mapping positions of these common reads were compared against both the CIGAR and FAT-CIGAR strings to obtain the following: the percentage of sequence reads that mapped to the exact same position against the reference with the exact same strings for all reference types, the percentage of reads that mapped to the exact same position but with different alignments, the percentage of reads that mapped to different positions but with the exact same alignment across all references and finally, the percentage of reads that mapped to different positions with different alignments.

3.2.3 Comparison of Variant Calling Accuracy Across Various Graph Genome software

In order to compare the accuracy of variants called from a wider range of currently available graph genome software, variant calling was also carried out using *BayesTyper* v1.5 (Sibbesen *et al.*, 2018) and *Graphtyper* v2.5.1 (Eggertsson *et al.*, 2017). As both the genome graphs and the read alignments against the graphs cannot be accessed, these software were omitted from the previous alignment comparison study. The variants called from *BayesTyper* and *Graphtyper* were compared against the variants called from the *vg* and *Graph Genome* toolkit reference and Bergstrom variation graphs, using the pipelines depicted in Figure 3.6. A limiting factor in the comparison with *BayesTyper* is that it is unable to be used for discovering novel variants as it is a genotyping toolkit therefore, unlike the other software, it is only able to genotype based on prior variants provided by the user.

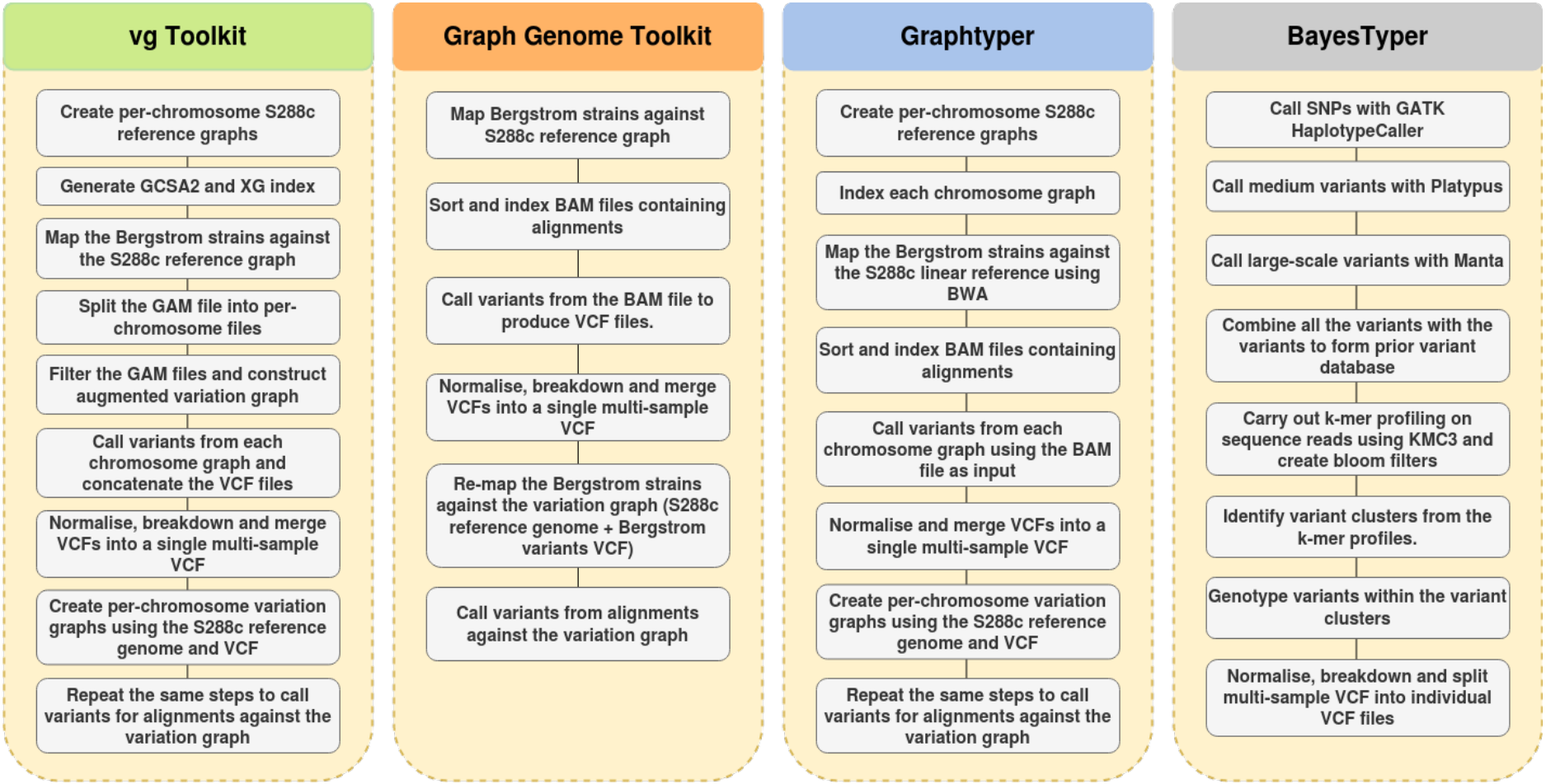


Figure 3.6: **Variant Calling Pipeline.** The flowchart shows the pipeline utilised in each software to align sequence reads against the graph and call variants.

Graphtyper

In order to construct the reference graph, the chromosome names within the S288c reference FASTA file were modified to numbers, as *Graphtyper* was developed to genotype human genomes, where FASTA headers are formatted as chromosome numbers. The sequence reads for all 19 strains were realigned against the modified FASTA file with *bwa mem* so that the BAM file contained the correct chromosome alignments. The BAM files were sorted, indexed with *SAMtools* and filtered to remove PCR duplicates using the *Picard MarkDuplicates* function. As *Graphtyper* only works on small regions of the genome, the *Graphtyper genotype* function was run to construct the reference graphs and call variants, limiting graph construction to each chromosome as with *vg*, using the *--region* parameter. The BAM file was utilised to identify regions containing variants to realign against the graph before variant calling. As *Graphtyper* also calls variants across small regions, this resulted in multiple VCF files output for each chromosome graph. The *bcftools concat* function was used to concatenate the VCF files across each chromosome before concatenating the 17 per-chromosome VCF files to produce a single VCF file for each of the 19 strains. The VCF files for each strain were normalised with *bcftools norm* and sorted with *bcftools sort*. In order to allow for cross-comparison with VCF files from the other software, *bcftools annotate* was run with the *--rename-chrs* parameter to change the chromosome names from, for example, '1' to 'S288c_ChrI' and *bcftools reheader* was used to change the sample names for the strains UWOPS03-461.4 and UWOPS83-787.8 which were abbreviated in the *Graphtyper* VCF file. The *bcftools merge* function was used to merge the VCF files into a multi-sample VCF file which was passed as input to the *genotype* function with the *--prior-vcf* parameter to construct the Bergstrom variation graph and call variants against the graph. The resulting VCF files were processed using the same methods as detailed for the reference graph.

BayesTyper

The initial step required combining the results of different variant callers in order to build a variant database from which the variation graph was constructed. The *BWA* alignments against the S288c linear reference genome for all 19 strains were used by *GATK HaplotypeCaller* v3.8.0 to call SNPs and small indels, *Platypus* v0.8.1 (Rimmer *et al.*, 2014) to call small to medium scale variants and *Manta* v1.5.0 (Chen *et al.*, 2016) to call large-scale variants. Prior to variant calling, *Picard* was used to add read

group information and read tags to the BAM header so that the BAM files could be processed with *GATK HaplotypeCaller*. PCR duplicates (reads that were obtained from the same DNA fragment) were also filtered and removed using the *Picard MarkDuplicates* function. Quality control is carried out in *Manta* before variant calling by generating alignment statistics to check whether over 90% of the reads are in normal read pair orientation. Due to low sequence quality, YJM981 failed quality control as only 81% of read pairs in the BAM file had the correct orientation. This error was bypassed by providing the alignment statistics XML file containing the fragment length distribution of the reads to *Manta*. The fragment length distribution was calculated using the *Picard CollectInsertSizeMetrics* function and the output text file was converted into an XML file using a custom Python script. The XML file was used as input for *Manta* which prevented it checking the read pair orientation.

For the prior variant database, initially the gVCF file containing variants from 1,011 *S. cerevisiae* strains (Peter *et al.*, 2018) from the 1002 Yeast Genomes Project was obtained from the project website (The 1002 Yeast Genomes Project, 2019). Any sites in the gVCF file without at least two alleles were filtered out in order to remove all the non-variant sites, and the file was converted into a VCF file using the *GATK GenotypeVCFs* function. However, as the downloaded gVCF file did not contain variants from the mitochondrial genome, variants were called from the mitochondrial genome using the same methods as outlined in the original paper. The sequences for 949 strains from the study were made publicly available on the NCBI Sequence Reads Archive (SRA) (Leinonen *et al.*, 2011). The strain sequence FASTQ files were downloaded using the SRA Toolkit v2.9.6 (accession no: ERP014555) and aligned against the S288c reference genome with *bwa mem*. The resulting BAM files were filtered to keep only these reads aligned against the mitochondrial genome, with any PCR duplicates removed. Local realignment was carried out using *GATK* to correct for mapping errors as mappers usually consider each sequence read individually and the scoring systems utilized may limit its ability to map well against indels. Mappers tend to favour mismatches and soft-clips instead of opening gaps in order to produce more optimal scoring alignments, which can lead to non-parsimonious mapping of reads against indels. As local realignment considers all the reads spanning the indel region, this allows for more high scoring, parsimonious alignments. *GATK RealignerTargetCreator* was used to identify target regions in which alignment could be improved and *GATK IndelRealigner* carried out

local realignment in the specified regions using a consensus model to produce realigned BAM files. *GATK HaplotypeCaller* was used to call variants from the mitochondrial genome using the realigned BAM files. All 949 VCFs were joined together using *GATK CombineVariants* and merged with the 1,011 strain VCFs using *GATK CatVariants* to form a multi-sample VCF file containing 1,935,064 variants. Due to the small size of the *S. cerevisiae* genome, provision of a large variant database prevented genotyping as *BayesTyper* was unable to estimate the parameters for the negative binomial genomic k-mer count distribution due to the absence of any k-mers that are unique to regions of the genome with no variation. As a result, a prior variant database was not provided as input.

The VCF files from *GATK*, *Platypus* and *Manta* for all 19 Bergstrom strains were combined using the *BayesTyperTools combine* function. The *KMC3* v3.0.0 (Kokot *et al.*, 2017) software was used to generate k-mer profiles by counting the k-mers from the FASTQ sequence reads to produce a compact representation of all unique and non-unique reads along with the frequency of occurrence for each strain. The *BayesTyperTools makeBloom* function was used to generate read k-mer bloom filters from the k-mer profiles. Variant candidates were identified using *BayesTyper cluster* based on the variants called using *GATK HaplotypeCaller*, *Platypus* and *Manta* and the output database from *KMC*. The variants within the identified clusters were genotyped using *BayesTyper genotype* to produce a single VCF file containing the genotyped variants from all 19 strains.

3.2.4 Simulation Study to Compare Graph Genome Software

As a benchmark dataset is required in order to truly compare the accuracy of the variant calls from each graph genome software, simulated benchmark VCFs were generated by evolving the Bergstrom strains using S288c chromosome I (S288c_ChrI) as the ancestral sequence and attempting to identify the variants induced within the simulated genomes. As simulating the S288c whole-genome would be computationally both time and memory expensive, Chromosome I was selected as it was previously found to display the second highest degree of intragenic variation at the nucleotide level after the mitochondrial genome amongst the Bergstrom strain population.

Seq-Gen

The distance matrix for the neighbor-joining SNP tree of the *S. cerevisiae* strains was obtained from Gianni Liti, senior co-author of both the Bergstrom study and the 1002 Yeast Genomes Project. As the SNP distance matrix contained 39 strains in total, any additional strains other than the Bergstrom strains and the S288c reference genome were removed from the matrix. The distance matrix was formatted as a tab-separated file containing the number of strains (20) and the name and SNP distances as a lower matrix and used as input for the *PHYLIP* v3.697 (Felsenstein, 1989) neighbor program to create a new neighbor-joining tree.

Initially, *Seq-Gen* v1.3.4 (Rambaut and Grass, 1997) was used to evolve 1,000 simulations of the Bergstrom strains along the SNP tree using the markov General Time Reversible (GTR) model (Tavaré and Miura, 1986) for nucleotide substitution. The GTR model was chosen instead of the Hasegawa, Kishino and Yano (HKY) model (Hasegawa *et al.,* 1985) and the F48 model (Felsenstein and Churchill, 1996), both of which model under the assumption that transitions and transversions occur at different rates, as it is a more complex model that assumes different rates of substitutions for each of the six nucleotide pairs. It also models under the assumption that each nucleotide occurs at different frequencies and that a base can evolve before changing back to its original state (also known as time reversible) i.e. C -> G -> C (Tavaré and Miura, 1986). The input file for *Seq-Gen* contained the following information: the number of sequences (1), the length of S288c_ChrI (230,218 bp), the S288c_ChrI sequence and the Bergstrom SNP tree output by *PHYLIP* in Newick format. The *Seq-Gen* run produced 1,000 multi-FASTA files, each containing the simulated genome sequences for the 19 strains which were split into individual FASTA files. In order to identify the exact variants induced within each simulated genome, pair-wise alignment was carried out using both *MUMmer* v3.23 (Kurtz *et al.*, 2004) and *progressiveMauve* vsnapshot_2015_02_13 (Darling *et al.*, 2010) between each of the 19,000 simulated genomes and the S288c_ChrI reference sequence. The *nucmer* program from the *MUMmer* suite was used as it allows for variant detection from the alignments and outputs the variants as a delta file. The *progressiveMauve* program was run with the *--collinear, --seed-family* and *--disable-backbone* parameters which uses spaced seeds to improve seed sensitivity during alignment within regions of low sequence identity. The simulated genomes were found to be unable to align with both *progressiveMauve* and *nucmer* as the simulated

genomes evolved with *Seq-Gen* were highly divergent due to S288c_ChrI not being specified as the ancestral node within the input tree. As the *-k* parameter within *Seq-Gen* automatically forces the first node on the tree to be selected as the ancestral node, which cannot be changed, the *nw_reroot* function from the *Newick Utilities* v1.6 (Junier and Zdobnov, 2010) toolkit was used to re-arrange the nodes in order to produce a Newick tree with S288c_ChrI as the starting node. *Seq-Gen* was re-run with the re-rooted tree and the *-k* parameter and pair-wise alignment was carried out on the output FASTA files. However, 1,087 and 1,088 of the simulated sequences were still found to be unable to align against the reference with *nucmer* and *progressiveMauve*, respectively, as these simulated genomes were still quite highly diverged from the reference sequence. 948 out of the 1,000 simulated genomes from W303, the most distant strain to S288c, were found to be unable to align in both aligners. As scaling both the branch and tree lengths within *Seq-Gen* did not improve alignment, it was necessary to find a pipeline in which the level of divergence, especially within the more distant strains, could be controlled to mimic the real Bergstrom strains.

TreetoReads

The *TreetoReads* v0.0.4 (McTavish *et al.*, 2017) software was utilised instead within the simulation study. It essentially carries out the same pipeline as that in the previous section in that it first simulates genomes down an input tree with *Seq-Gen* using the S288c reference sequence itself as a leaf in the tree. For each simulated genome, it outputs a FASTA file containing the genome sequence and FASTQ files containing paired-end Illumina sequence reads simulated from the genome sequence using the *ART* vART-GreatSmokyMountains-04-17-2016 (Huang *et al.*, 2012) software. The input configuration file for *TreetoReads* was prepared with the following parameters: the Bergstrom SNP tree, the number of variable sites (20,000), the S288c_ChrI reference FASTA file, the nucleotide substitution rate matrix (AC: 0.94614, AG: 1.15179, AT: 1.90661, CG: 1.75553, CT: 2.14488, GC: 1), read coverage (30x), the percentage of mutation clustering (0.05), exponential mean (125), the indel length distribution model (LAV 1.7 541), indel rate (0.1), the *ART* error profile, read length (108bp), fragment size (380) and the standard deviation for the fragment size (120). *TreetoReads* utilises *Seq-Gen* to simulate genomes and the *INDELible* v1.03 (Fletcher and Yang, 2009) software to simulate indels within each genome. As the SNP tree from *PHYLIP* contained

negative branch lengths which *INDELible* was unable to process, the SNP tree was re-created with *MEGA X* v10.1 (Kumar *et al.*, 2018) to ensure all of the branch lengths were positive before the tree was re-rooted with *Newickutils nw_reroot* to make S288c the first tip on the tree.

The *art_profiler_illumina* function from the *ART* software was used to estimate the error profiles based on the observed data from the original Bergstrom FASTQ files. The error profiles are utilised to simulate sequence reads using an empirical error model by taking into account the base quality score distribution (Huang *et al.*, 2012). The fragment size and read length value was also determined using the longest read length observed within the error profiles. The relative rates of nucleotide substitution were estimated from the actual Bergstrom strains using the *PAUP* v4.0a168 (Swofford, 2002) software. In order to obtain the substitution rates for S288c_ChrI, reads that mapped to S288c_ChrI were obtained from the BAM files containing alignments output by *BWA* for all 19 Bergstrom strains and both the read names and sequences were parsed into FASTA file format using *awk*. Multiple sequence alignment of the FASTA files was carried out with *progressiveMauve* to produce an extended multi-FASTA (XMFA) file. This was converted into FASTA format using the xmfa2fasta perl script (https://github.com/kjolley/seq_scripts.git) and converted to the NEXUS file format using *PAUP* to estimate the substitution rate matrix. The *jModelTest2* v2.1.10 (Darriba *et al.*, 2012) software was utilised to check how the nucleotide frequency across the Bergstrom strains differs from S288c. The nucleotide rates were obtained from the alignment FASTA files under the following four criteria using the parameters: *-BIC (Bayesian Information Criterion), -AIC (Akaike Information Criterion), -AICc (corrected Akaike Information Criterion), -DT (Decision Theory criterion), -v (model averaging) and -a (estimate model-averaged phylogeny).* The average nucleotide rates for the Bergstrom strains were as follows: A: 0.30, C: 0.20, G: 0.20, T: 0.30, which were found to be highly similar to those for S288c: A: 0.29, C: 0.21, G: 0.21, T: 0.29.

The number of SNPs in S288c_ChrI of each of the Bergstrom strains was obtained by averaging the variant calls from chromosome I of the *vg* reference graph, variation graph and *BWA*. A trial and error process was used in which *TreetoReads* was run several times with different numbers of variable sites to identify the number of variable sites where both the number of SNPs and the number of shared SNPs across the simulated genomes

mimicked the real Bergstrom strains. As the more distant strains were highly divergent and had an over 10-fold increase in the number of SNPs induced by *TreetoReads*, the tree branch lengths for these strains were required to be scaled down. Initially, the *TreetoReads* Python script was edited to scale down the branches of the distant strains. However, this also resulted in changes to the branch lengths of other strains and scaling down the branch lengths any greater than 100 times did not result in further reduction to the number of SNPs, which was vastly higher than the expected range. Therefore, the distance between the branches were also manually scaled down between 10-100 times for the following strains to obtain the final input tree: BC187, YJM978, YJM975, YJM981, DBVPG1373, DBVPG1106, DBVPG6765, DBVPG1788, L1374 and L1528.

Once the optimal parameter settings had been identified, *TreetoReads* was run 1,000 times to obtain 1,000 datasets, each containing the FASTA and FASTQ files for 19 simulated genomes. For each genome, the FASTA file was aligned against the S288c_ChrI reference with *nucmer* to obtain the alignment delta files and *show-snps* was used to obtain the variants from the alignment. The *my-mummer-2-vcf.py* Python script (https://github.com/MatteoSchiavinato/Utilities/blob/master/my-mummer-2-vcf.py) was used to convert the SNP files into VCF files generating the truthsets required to carry out variant comparison across the four graph genome software.

Variant Calling

BayesTyper, *Graphtyper* and the *Seven Bridges Graph Genome* toolkit were run on all 1,000 simulated datasets using the same methods as previously outlined above to produce variant calls against both the S288c_ChrI reference graph and the Bergstrom variation graph. As there had been several upgrades to the *vg* toolkit since the earlier analysis, *vg* 1.26 was utilised for the simulation study. The reference and Bergstrom variation graphs was constructed, indexed without pruning and reads were aligned using the same methods as described for *vg* 1.18. In a deviation from the *vg* 1.18 methodologies due to the software update, after filtering the GAM alignment files with *vg filter*, *vg augment* was run with the following parameters to generate the augmented graph without computing read support: *-s* (computes subgraphs), *-m 4* (minimum coverage of reads required at breakpoints to be added to the graph), *-q 5* (minimum base quality), *-Q 5* (minimum mapping quality of read) and *-A* (augmented GAM file). The

augmented graph was indexed with *vg index* and read support was computed from the augmented GAM file using *vg pack* to produce a PACK file. Variant calling was carried out by running *vg call* with the XG index of the augmented graph and *-k* to specify the PACK file.

In order to compare the various graph genome software against the conventional linear pipeline for variant calling, all 19,000 simulated genomes were aligned against the S288c_ChrI reference sequence with *bwa mem* and *FreeBayes* v1.3.2 (Garrison and Marth, 2012) was used to call variants from the alignments. The VCF files were normalised with *bcftools norm* and sorted with *bcftools sort*, as had been carried out for all of the graph genome software.

In order to compare the variant calls generated from each pipeline against the truthset, the number of SNPs, indels and total variants were parsed out of the VCF files for each pipeline to evaluate how similar the number of calls were to the truthset. *R* was used to produce dot plots comparing the number of variants against the truthset, which revealed several outlier points demonstrating both over and under-calling of variants in both the *vg* and *BayesTyper* variant calls. The outlier datasets were re-run in *vg* to check whether the variant calls were reproducible. However, when re-run the variant calls were not reproducible and variants calls in all of the re-run datasets were in the expected range when comparing against the truthset and the variant calls from the other pipelines. This suggested a fault in the initial *vg* runs which was speculated to be caused by memory issues in *vg* that constantly caused the runs to crash when the datasets were run in parallel using multiple threads. In order to overcome the memory issue, all 1,000 datasets were re-run in *vg* serially and the number of variants were again parsed out to ensure there were no longer any outlier datasets. The first dataset (Set 1) was also re-run 10 times through all of the graph genome software in order to identify whether any of the variant callers are deterministic. Set 1 was run a further 1,000 times as single runs calling variants against the *vg* reference graph to confirm that the *vg* algorithm is stable. This method of comparing the number of variants against the truthset, however, was determined to be highly inaccurate as it did not account for both the differences in variant representation between each variant caller and the compounding of proximal variants preferred by some variant callers to form more complex calls (i.e. haplotype calls).

Vcfeval Classification of the Variants

In order to mitigate this issue, the *RealTimeGenomic (RTG) Tools* v3.12 (Cleary *et al.*, 2015) *rtg vcfeval* command was utilised as it is able to easily identify the same variant call, even if there are discrepancies in variant representation between different variant callers, for both simple and highly complex variants. It compares each called VCF file against the truthset VCF file and outputs three individual VCF files containing true positive, false positive and false negative calls. Prior to running the variant call comparison, the S288c_ChrI FASTA file was required to be converted into Sequence Data Format (SDF) using *rtg format* which converts the FASTA sequence into multiple binary files so that it can be processed efficiently. The *rtg vcfeval* command was then run with the truthset VCF file, the called VCF file and the SDF file using the *--all-records* parameter to specify that all of the records within the VCF file should be evaluated. The *vcfeval* comparison was run separately for variant calls against the reference and Bergstrom variation graphs from *vg*, *Graphtyper* and the *Seven Bridges Graph Genome Pipeline*.

In order to ensure that the output from *vcfeval* was accurate, the variants within the true positive VCF files from the first dataset for all of the pipelines were compared by visualising the alignment BAM files from the *Graph Genome Pipeline* with *SAMtools tview* to manually verify the variants. Manual inspection of the variant calls identified errors in the truthset arising from incorrect local alignment of the genomes in *nucmer*. Therefore, the truthset was re-generated by running *pIRS* to simulate long paired-end reads from the FASTA reference sequence of the simulated genomes output by *TreetoReads* to produce accurate alignments. Sequence reads of length 1,000bp and 50x read coverage were generated using *pirs simulate* and the following parameters were used to increase sensitivity for variants with less read support and ensure there were no sequencing errors simulated within the reads: *--error-rate=0*, *--no-substitution-errors*, *--no-indel-error*, *--no-gc-bias*, *-min-alternate-count 1* and *min-alternate-fraction 0*. The simulated reads were mapped against the S288c_ChrI reference sequence with *BWA* before variant calling with *FreeBayes* to generate the ground-truth VCF files for all of the datasets. As inspection of the *vg* variant calls demonstrated further issues with the variant calling algorithm. Therefore, *vg* v1.18, which was utilised for the alignment comparison study, was also run on all 1,000 simulated datasets using the same methods to identify whether the previous versions also suffered from the same issues. The VCF files from

all six pipelines and the truthset were further decomposed using *vcflib* v1.0.1 (Garrison *et al.*, 2021) to allow for easier comparison by *rtg vcfeval*. The *vcflib vcfbreakmulti* function was used to break down any haplotype calls from haplotype callers such as *vg* and *vcfallelicprimitives* was used to break down complex calls into several individual variant calls. The decomposed VCF files were further sorted with *vcfstreamsort* and duplicate variants were removed using *vcfuniq* to produce the final post-processed VCF files from all of the pipelines on which *rtg vcfeval* was re-run.

The *rtg vcfeval* function was found to misclassify a few variants even when the variant representation was the same in both the called VCF file and the truthset VCF file. Thus, a custom Python script (https://github.com/prithikasritharan/VcfevalFilter/blob/main/README.md) was written to correct any misrepresented variants. For each of the 19,000 simulated genomes, the Python script read the truthset VCF file and the processed VCF files from each of the pipelines into memory. For each pipeline, the script also read the false positive VCF file output by *rtg vcfeval* and compared the reference allele, alternate allele and position against the variants in the truthset VCF file. If records were observed within the truthset that matched exactly for all three fields, this identified the variant as a true positive (TP) call. If a matching position was not found for the record, variants that may have been represented differently were identified from the truthset VCF file by searching for variants which occurred in regions spanning twice the variant allele length on either side of the variant position. The position, reference allele and alternate allele sequences were compared for each identified variant in the truthset to determine true positive variants that had been misclassified due to differences in representation. A corrected false positive (FP) VCF file was written out without the variant and a corrected TP VCF file was written out with the variant added. The script also read the false negative (FN) VCF file and compared the three fields against the variants within the called VCF file. Any variants that were present in the called VCF file were removed from the corrected FN VCF file and added to the corrected TP VCF file. The script also annotated the format field of the truthset according to whether each variant was a TP or FN. Variants in the called VCF file with either a TP or FP annotation were input to the *rtg vcf2plot* function. This was used to calculate the true positive and false positive rates based on the genotype quality (GQ) score (the phred quality score assigned by the variant caller to determine the certainty of a genotype assertion) to produce receiver-operating curves comparing each pipeline. However, the

receiver-operating curves could not be produced as the *rtg vcf2plot* function was unable to read the GQ score within the VCF files from both *vg* 1.18 and *vg* 1.26 even though they were present. The number of variants in the TP, FP and FN VCF files for all of the pipelines were parsed out and *R* was utilised to compare the variant calls.

Analysis of the Variant Calls

The preliminary analysis of the datasets also showed a greater number of false positive variants in *vg* 1.18 variant calls, even though the *vg* aligner was found to perform better mapping of the sequence reads in the previous sections. In order to verify that the performance of the *vg* variant caller was not due to that specific version and to check the stability of the earliest version of *vg* used in this project, *vg* 1.5 was run on all 1,000 datasets to call variants against both the reference and Bergstrom variation graphs using the same methods as previously outlined in Chapter 2. Unlike *vg* 1.18 and *vg* 1.26, the *vg* 1.5 Bergstrom graph had to be pruned prior to the GCSA2 index generation as indexing in the earlier versions of *vg* is a computationally memory intensive process. Pruning was carried out using the *vg prune* function in vg1.9 as running the *vg mod* function in *vg* 1.5 produced a software assertion failure error due to a bug in the code.

Finally, as the thresholds utilised for variant calling were different across each of the graph genome software, the variant calls had to be filtered to cut-off any variants without a minimum allele fraction >= 0.2. The minimum allele fraction, m, is defined as

$$m = \frac{\text{no. of observations of the allele}}{\text{read depth at that position}} \tag{3.3}$$

The cut-off value was determined by comparing the *vg* variant calls from both the reference and Bergstrom variation graphs thresholded at 0.2 and 0.9 for the first five datasets. In *vg* 1.5 and *vg* 1.18, filtering was carried out by re-running *vg call* with the *-m* parameter. In *vg* 1.26, the *-m* option was no longer present and instead *vg call* was run with the *--min-support* parameter in which both the minimum no. of observations of the allele required and the minimum no. of reads support at that site was specified as each of the following in turn: (2, 5); (4, 5); (2, 10); (9, 10). As these values are not scaled, even if the site support increases, the minimum no. of observed alleles remains the same which may still allow for variants with low read support to be called. As comparison of the variant calls demonstrated that the 0.9 cut-off value was

too stringent, therefore, 0.2 was chosen as the cut-off to allow for false positive variants with low read support to be filtered without discarding the true positive variants.

Variant calling was re-run for the reference graph for all 1,000 datasets from *vg* and *Graphtyper*, thereby carrying out the filtering before post-processing the VCF files and re-running *rtg vcfeval*. As the Bergstrom variation graphs were constructed using the unfiltered variant calls, they were likely to incorporate false positive variants within the reference that allowed for miscellaneous alignments resulting in a false positive bias. In order to overcome any false positive bias, the Bergstrom variation graph for all of the datasets had to be re-constructed from the filtered variant calls before repeating the process of mapping and variant calling. For *vg* 1.26 and *Graphtyper*, the variants were filtered from the VCF files after variant calling using *bcftools annotate* to filter on the AD (allele depth) and DP (read depth) values in the FORMAT field.

As the *Graph Genome Pipeline* already thresholds m at 0.2 during variant calling, it did not require any further filtering. There is no information available regarding the thresholds used for variant calling in *BayesTyper* nor does the VCF files contain any information pertaining to the allele or read depth therefore the *BayesTyper* variant calls could not be filtered. The filtered VCF files were run through the post-processing pipelines, re-classified with *rtg vcfeval* and the classifications were further corrected with the custom Python script. The number of variants predicted by all pipelines were parsed out and the most accurate software for variant calling from the simulated dataset was determined by calculating the precision, recall and F1 scores, the latter of which accounts for the trade-off between precision and recall, using the *R* software. The precision, recall and F1 scores were defined as follows:

$$Precision = \frac{\text{no. of true positives}}{\text{no. of true positives} + \text{no. of false positives}} \tag{3.4}$$

$$Recall = \frac{\text{no. of true positives}}{\text{no. of true positives} + \text{no. of false negatives}} \tag{3.5}$$

$$F1\ score = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.6}$$

3.3 Results

3.3.1 *FAT-CIGAR* Toolkit Run Time

The performance of the *FAT-CIGAR* toolkit, specifically in regard to the impact of the size of a dataset on the run time, was monitored for alignments from *BWA* and the *vg* reference graph using the Bergstrom strains, as shown in Table 3.2. The *FAT-CIGAR* toolkit was found to run notably faster on the *BWA* alignments, as expected, than the *vg* reference graph for all 19 strains. The Hawaiian mosaic strain, UWOPS87-2421, which contains around 84 million sequence reads, took one hour and 45 minutes to run for alignments from *BWA* whereas it took over 22 hours to run for alignments from *vg* with the initial index generation time taking over three hours for a 136Gb JSON file. Excluding UWOPS87-2421 and UWOPS83-787.3, the average script run time taken to construct the FAT-CIGAR string for alignments from *vg* was around 22 minutes, whereas, for linear alignments, it took around three minutes. Overall, for both references, a huge increase in the number of sequence reads results in an exponential increase in run time.

3.3.2 Comparison of Sequence Read Mapping Across Different Graph Genome Software

The percentage of trimmed sequence reads from the Bergstrom strains that aligned against the reference graph and variation graph for *vg* and the *Graph Genome* toolkit and the linear reference genome for *BWA* has been shown in Figure 3.7. The line graph shows that a greater percentage of reads that were able to align to both the variation and reference graphs from *vg* and the *Graph Genome* toolkit than in comparison to the linear reference genome. The *vg* variation graph was found to perform best in aligning the greatest percentage of sequence reads, followed by the *vg* reference graph. Both *vg* graphs performed better than the *Graph Genome* toolkit graphs in terms of aligned reads.

Fourteen strains were able to align better against the *vg* variation graph and 13 strains against the *vg* reference graph compared to the *Seven Bridges* variation graph in which only four strains aligned better than the *vg* variation graph and two strains better than the *vg* reference graph. The percentage difference in mapping between

Table 3.2: ***FAT-CIGAR* Toolkit Run Time.** The script run time for alignments from both *BWA* and the *vg* reference graph for the Bergstrom strains.

Strain	**No. of Reads**	**Run Time (hh:mm:ss)**	
		vg	***BWA***
UWOPS87-2421	84,333,836	22:07:36	01:45:41
UWOPS83-787.3	66,433,124	15:13:59	01:23:18
YPS128	5,589,034	50:11	07:15
SK1	3,725,974	34:23	04:53
DBVPG6765	3,549,980	35:34	04:23
W303	3,290,148	28:58	03:57
DBVPG6044	3,106,876	27:55	03:42
DBVPG1106	3,080,300	26:36	04:01
L1528	2,923,432	24:35	03:39
Y12	2,656,714	23:20	03:25
Y55	2,655,346	23:50	03:29
DBVPG1373	2,604,596	22:02	03:37
UWOPS03-461.4	2,019,656	17:59	02:41
YJM975	1,955,320	17:17	02:30
L1374	1,563,798	13:52	01:58
BC187	1,417,092	11:58	01:52
YJM978	1,310,934	13:22	01:43
DBVPG1788	1,266,912	10:39	01:34
YJM981	353,384	02:55	00:29

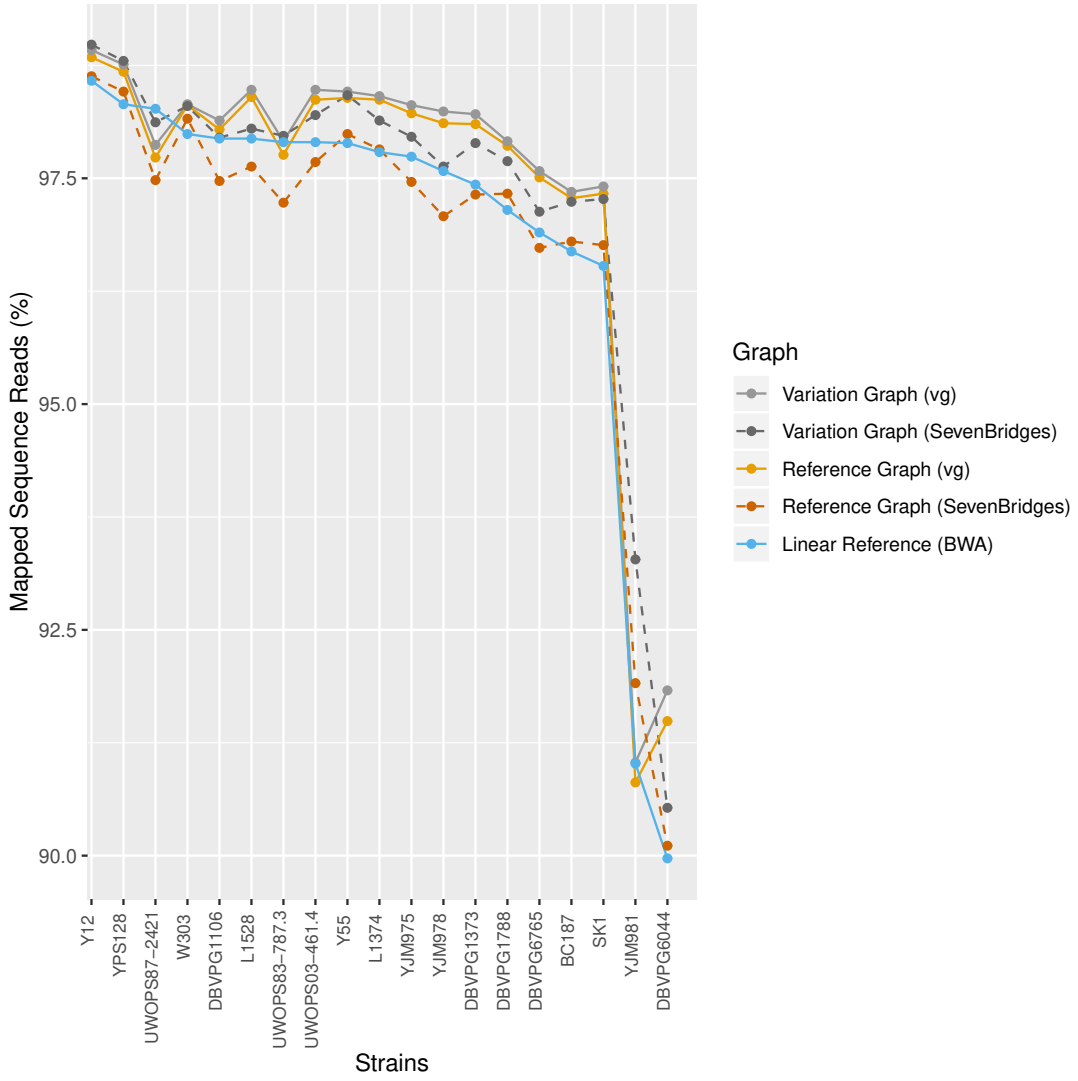


Figure 3.7: **Trimmed Sequence Read Mapping of the Bergstrom strains.** This figure shows the percentage of sequence reads that aligned against the *vg* variation graph (light grey solid line), the *Seven Bridges* variation graph (dark grey dotted line), the *vg* reference graph (light orange solid line), the *Seven Bridges* reference graph (dark orange dotted line) and the S288c linear reference genome (blue solid line).

the *vg* variation graph and *vg* reference graph was found to be 0.01-0.34%, 0.02-2.25% between the *vg* variation graph and the *Graph Genome* toolkit variation graph and 0.16-1.72% between the *vg* variation graph and the *Graph Genome* toolkit reference graph. The *Graph Genome* toolkit reference graph was found to perform similarly to *BWA* with 9 strains having the lowest percentage of reads able to align against the reference graph and 10 strains the lowest against the linear reference genome, with a 0.03-0.89% difference in mapping. *BWA*, however, was shown to have aligned the highest percentage of reads for UWOPS87-2421, 98.27%. The average percentages of

mapped reads across the five reference structures for all 19 strains were as follows: *vg* variation graph - 97.45%, *vg* reference graph - 97.35%, *Graph Genome* toolkit variation graph - 97.34%, *Graph Genome* toolkit reference graph - 96.84% and *BWA* - 87.64%.

3.3.3 Comparison of Sequence Identity Scores Across Differently Evaluated Read Alignments

Sequence identity scores were calculated for both the non-surjected CIGAR and FAT-CIGAR strings generated using the *FAT-CIGAR* toolkit for alignments from all five reference structures for the 19 Bergstrom strains. The histogram in Figure 3.8 shows the sequence identity scores for the strain DBVPG1106 (see Appendix Figure B.1 to B.18 for histograms for the other 18 strains). For all five reference structures, there is shown to be a decrease in the number of perfectly mapped reads, reads that have a sequence identity score of one indicating the likelihood of a true alignment, from the CIGAR string to the FAT-CIGAR string.

Table 3.3: **Comparison of Sequence Identity Scores for DBVPG1106.** The percentage of sequence reads that had perfect mapping (sequence identity score 1) and the percentage of reads that had a sequence identity score greater than 0.9 across the five reference structures, for both CIGAR and FAT-CIGAR strings.

Reference	**CIGAR**		**FAT-CIGAR**	
	Perfect Mapping	**Score >= 0.9**	**Perfect Mapping**	**Score >= 0.9**
vg variation graph	**94.9**	96.0	**75.3**	95.8
vg reference graph	91.6	95.1	53.4	94.7
GGP variation graph	92.8	**96.6**	74.2	**96.4**
GGP reference graph	85.3	94.5	53.3	94.0
Linear reference	92.0	95.7	53.5	95.3

The differences in the proportion of perfectly mapped reads and well-mapped reads

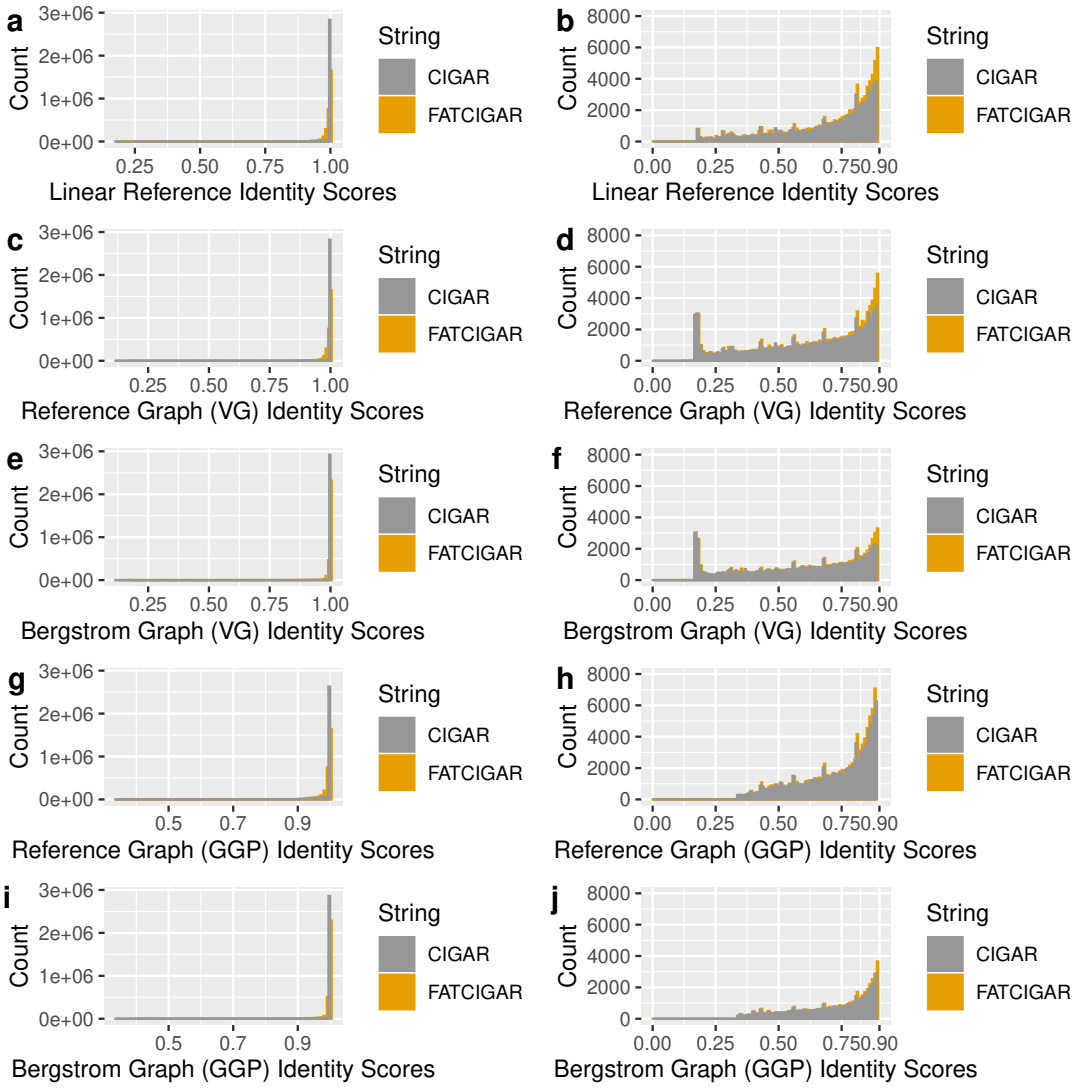


Figure 3.8: **Sequence Identity Scores for DBVPG1106.** This histogram shows the sequence identity scores between 0 and 1 (left) and between 0 to 0.9 from the same histogram (right) for a & b) the linear reference genome, c & d) the *vg* reference graph, e & f) the *vg* Bergstrom graph, g & h) the *Graph Genome* toolkit (GGP) reference graph and i & j) the *Graph Genome* toolkit variation graph.

(reads with sequence identity scores >=0.9) when comparing the CIGAR and FAT-CIGAR strings has been shown in Table 3.3. Approximately 53-75% of perfectly mapped reads according to the CIGAR string were also perfectly mapped according to the FAT-CIGAR string. There was only a 0.2-0.5% decrease in the proportion of well-mapped reads from the CIGAR to the FAT-CIGAR string. The variation graphs, for both *vg* and the *Graph Genome* toolkit, had the highest percentages of perfectly mapped reads, in comparison to the two reference graphs and linear reference genome. Also, there was not as great a decrease in the number of well-mapped reads and reads with perfect mapping from the CIGAR to the FAT-CIGAR string in the variation graphs. The linear reference genome had more reads with a greater identity score compared to the reference graphs. The FAT-CIGAR sequence identity scores showed that the *vg* reference and variation graph were able to align a greater proportion of sequence reads with perfect mapping in comparison to *Graph Genome* toolkit. However, the *Graph Genome* toolkit variation graph was found to align the greatest proportion of well-mapped reads across all five reference structures.

3.3.4 Comparison of Read Alignments Against Mapped Locations

The mapped locations of common read alignments across all five reference structures were compared against both the CIGAR and FAT-CIGAR strings for the 19 Bergstrom strains, as shown in Figure 3.9. The majority of the sequence reads mapped at the same position with the same CIGAR string (74-83% across the Bergstrom strains, except YJM981 with only 40%, the latter result likely due to poor read quality reducing the accuracy of mapping) and the same FAT-CIGAR string (31-77%). The percentage of sequence reads that had the same CIGAR string was found to be higher for each strain than the percentage of reads with the same FAT-CIGAR string. Reads that mapped to different positions with the same CIGAR string was shown to decrease slightly, by 0.9-3.2%, for the FAT-CIGAR string. There was also an increase of 4.6-32.2% in the reads that mapped to the same position with different CIGAR string (2.5-9%) to the FAT-CIGAR string (7.1-39.7%) due to the increase in perfect mapping, which is also reflected in the sequence identity histogram in Figure 3.8. Similarly, the sequence reads that mapped to the different positions with different CIGAR string (5.3-37.1%) were also shown to increase by 0.7-3.24% when compared to the FAT-CIGAR string (7.3-14.4%).

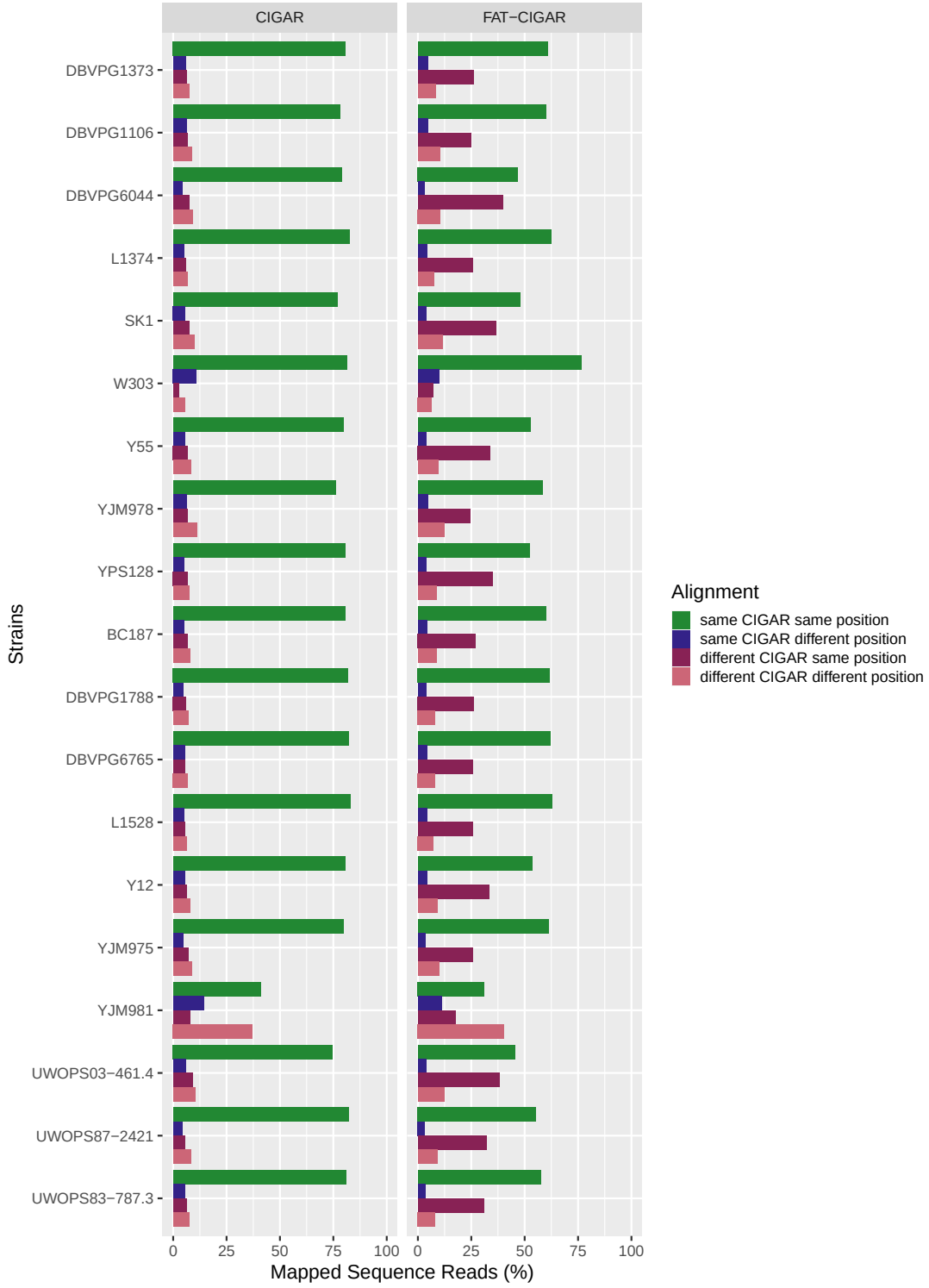


Figure 3.9: **Comparison of Alignments Against Mapped Locations.** This bar chart shows the percentage of sequence reads that mapped across all five reference structures with the same CIGAR or FAT-CIGAR string at the same position (green), same string at a different position (blue), different string at the same position (purple) and different string at a different position (pink).

3.3.5 Comparison of the Bergstrom Strains Variant Calling Across Different Graph Genome Software

The numbers of SNPs (see Table 3.4) and indels (see Table 3.5) called from each of the four graph genome software were compared for the Bergstrom strains. The reference graphs for *vg*, *Graph Genome Pipeline* and *Graphtyper* were found to call fewer SNPs and indels in comparison to the variation graphs. In *vg*, which had the greatest difference between the two references, there was an increase of between 1,542 to 14,144 SNPs and 833 to 4,001 indels between the reference and variation graphs per strain, with an average increase of 7,724 SNPs and 1,893 indels. For the *Graph Genome* toolkit, there was an increase of 594 - 2,742 SNPs and 124 - 402 indels (with an average increase of 1,522 SNPs and 284 indels) and for *Graphtyper*, there was an increase of 42 - 2,518 SNPs and 104 - 1,841 indels per strain (with an average increase of 1,108 SNPs and 655 indels). The *vg* reference graph had the least number of SNPs called for 13 of the strains and the least number of indels for eight of the strains, followed by *BayesTyper* which had the least number of SNP calls for five strains and the *Graph Genome Pipeline* reference graph which had the least number of indel calls for seven strains. The *vg* variation graph was also found to call the highest number of SNPs for all 19 strains and the highest number of indels for 15 strains. The overall average of the number of variants (SNPs and indels) called for each reference structure was as follows in ascending order: *vg* reference graph - 28,748, *BayesTyper* - 29,037, *Graph Genome Pipeline* reference graph - 29,390, *Graphtyper* reference graph - 30,069, *Graph Genome Pipeline* variation graph - 30,299, *Graphtyper* variation graph - 30,945 and *vg* variation graph - 33,527.

Strain	*vg* reference graph	*vg* Bergstrom graph	*GGP* reference graph	*GGP* Bergstrom graph	*Graphtyper* reference graph	*Graphtyper* Bergstrom graph	*BayesTyper*
DBVPG1373	45170	50924	45689	46887	47181	48623	46359
DBVPG6044	77980	85112	79401	80754	80694	81147	78725
L1374	43428	51256	44733	45957	46810	47164	45922
SK1	74232	81224	75348	77177	76380	77993	73393
UWOPS83-787	73130	76362	71674	72733	72673	73576	66017
W303	8562	13045	9755	10250	10174	10216	9853
Y55	61042	72684	66437	68131	68043	69368	65343
YJM978	41316	53524	44296	45879	47099	49617	45300
YPS128	67306	72416	67667	68946	68566	69252	65754
BC187	43914	56199	46219	47792	49187	50818	46834
DBVPG1106	44776	50208	45323	46732	46589	47220	46075
DBVPG1788	41732	53857	44248	45625	47510	48359	44991
DBVPG6765	45574	51845	45777	47395	47146	48884	46274
L1528	44771	50848	45411	46969	46998	47842	46335
UWOPS03-461	79728	93872	82360	84750	86092	87706	79529
UWOPS87-2421	73955	75497	72945	74252	73237	74593	67158
Y12	66077	79927	67536	70278	73246	74440	64472
YJM975	44213	50850	45112	46392	46848	47031	46204
YJM981	9367	13395	10307	12275	10164	11843	8327

Table 3.4: **Differences in Variant Calling Across the Graph Genome software.** The numbers of SNPs called from the 19 Bergstrom strains against the reference and Bergstrom variation graph using *vg*, the *Graph Genome* toolkit, *Graphtyper* and *BayesTyper*.

Strain	*vg* reference graph	*vg* Bergstrom graph	*GGP* reference graph	*GGP* Bergstrom graph	*Graphtyper* reference graph	*Graphtyper* Bergstrom graph	*BayesTyper*
DBVPG1373	4732	6125	5145	5409	4638	5076	5584
DBVPG6044	6561	7721	7743	7974	6621	7002	8156
L1374	5982	8770	4835	5109	4575	5015	5447
SK1	6361	7579	7403	7782	6425	6931	7655
UWOPS83-787	6564	7689	7280	7447	7166	7386	7098
W303	1257	2674	1444	1568	1283	1387	1470
Y55	5526	7198	6645	6959	5828	6670	6984
YJM978	11284	18450	4844	5197	4655	5048	5315
YPS128	5946	6831	6833	7013	6016	7024	7088
BC187	6868	10721	5063	5465	4802	5620	5385
DBVPG1106	4785	6202	5064	5350	4494	5053	5404
DBVPG1788	7113	11114	4869	5148	4623	5067	5250
DBVPG6765	4487	5780	5095	5386	4576	5653	5502
L1528	4480	5812	5041	5390	4552	5249	5506
UWOPS03-461	9022	11860	7917	8261	7061	8902	7488
UWOPS87-2421	6644	7527	7325	7487	7260	7539	7083
Y12	6160	7734	6842	7224	6286	7295	6399
YJM975	6836	9905	4997	5342	4576	5323	5478
YJM981	2566	3302	2213	2677	2533	2984	2266

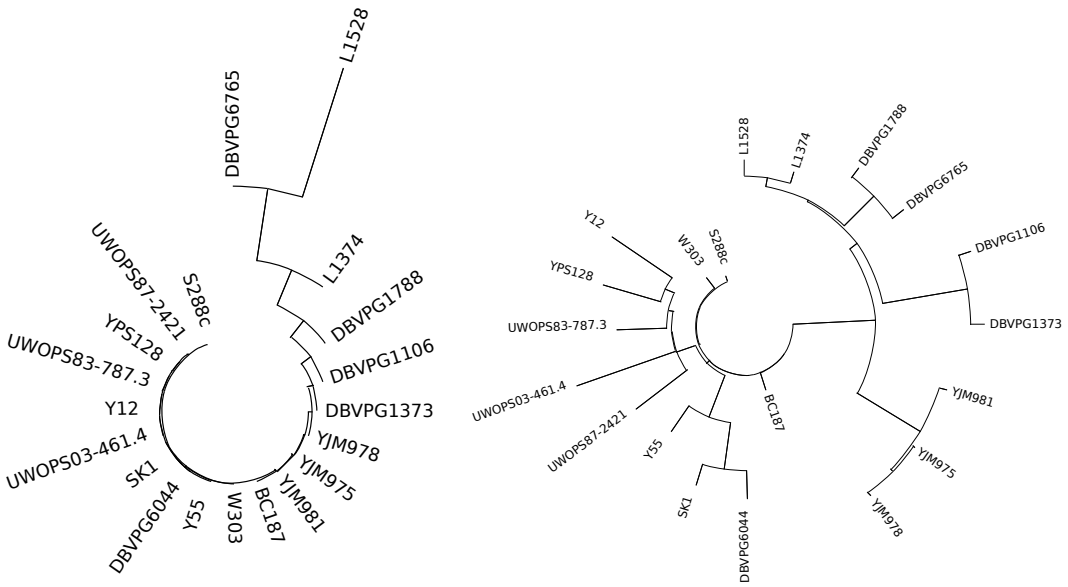
Table 3.5: **Differences in Variant Calling Across the Graph Genome software.** The numbers of indels called from the 19 Bergstrom strains against the reference and Bergstrom variation graph using *vg*, the *Graph Genome* toolkit, *Graphtyper* and *BayesTyper*.

3.3.6 Simulating Genomes with *TreetoReads*

Figure 3.10a shows the original whole-genome SNP tree of the Bergstrom strain population rooted on the S288c reference genome, from which the input tree for *TreetoReads* was re-constructed in *MEGA X*, to enforce positive branch lengths. The number of SNPs within chromosome I of the original Bergstrom strains was found to range from 226-3,560 SNPs, with an average of 1,680 SNPs and with 12 of the 19 strains possessing between 1,000-2,000 SNPs. In order for the simulated genomes to mimic the real strains, the number of variant sites input into the *TreetoReads* software was determined to be 20,000, as the resulting number of shared SNPs was highly similar to that of the real strains. Furthermore, nine of the simulated strains possessed between 1,000-2,000 SNPs, similar to the numbers seen in the real dataset. However, the high number of variant sites resulted in very high numbers of variants in some distant strains (i.e. 5,600 SNPs in YJM978 to 18,600 SNPs in L1528). As decreasing the number of variants sites would reduce the number of variants in the distant strains at the cost of greatly increasing the number of shared SNPs amongst the simulated genomes, the branches of the most distant strains were instead scaled down. The re-scaled tree was found to generate between 300 to 2,900 SNPs within the Bergstrom strains, with an average of around 1,900 SNPs per strain. The re-scaled tree, which could therefore be used to simulate SNPs and indels in similar numbers to that seen in the original dataset, is shown in Figure 3.10b.

3.3.7 Evaluation of the Deterministic Nature of the Variant Calling Algorithms

In order to evaluate whether the variant calling algorithms employed by *vg* v1.26, the *Graph Genome Pipeline*, *Graphtyper* and *BayesTyper* produced deterministic variant calls, the number of variants called was observed when the same dataset was run repeatedly in each pipeline. The *Graphtyper* variant caller was found to be stable as all 10 runs had the same number of variants called for the 19 strains (Appendix Table B.1). The *vg* and *Graph Genome Pipeline* variant calls were also found to be mostly stable with minor differences in the number of variants called across the different runs for a few strains. YPS128 was found to be the only strain with a one variant difference across the runs in *vg*, with 2,584 variants were called in eight runs and 2,585 variants in two runs (Appendix Table B.2). For the *Graph Genome Pipeline*, the strains DBVPG1106



(a) Original SNP Tree

(b) TreetoReads Input Tree

Figure 3.10: **SNP Tree of the Bergstrom Strains.** This figure shows (a) the original SNP tree of the Bergstrom strain population and (b) the SNP tree produced by *MEGA X* with re-scaled branches to mimic the number of shared variants within the Bergstrom strains.

and DBVPG6765 both had five runs with a one variant difference and DBVPG1788 had eight runs with 3,698 variants and two runs with 3,700 variants (Appendix Table B.3). The *BayesTyper* variant calling algorithm was found to be non-deterministic as the number of variants called was highly varied across the 10 runs for all 19 strains. The number of variants called ranged from a difference of three variants to 25 variants across the runs with only two runs on average producing the same numbers of variant for each strain (Appendix Table B.4).

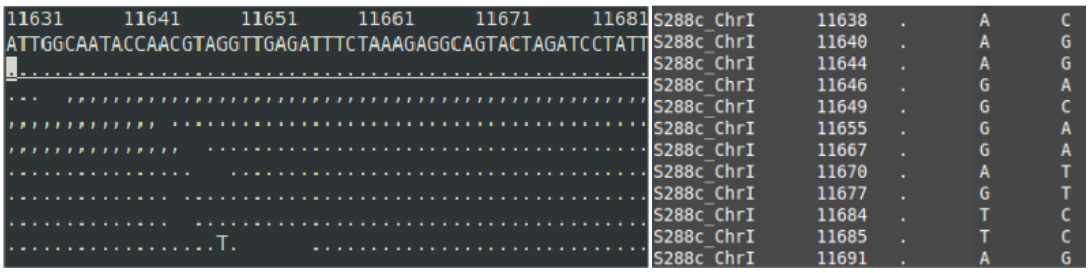
The number of variants called against the *vg* reference graph across a further 1,000 repeated runs on Set 1 has been shown in Table 3.6. Fourteen strains had the same number of variants called across all 1,000 runs whilst five strains had slight differences in the variant calls in four or fewer runs. Both SK1 and L1528 had one run with a single variant difference, UWOPS03-461.4 had three strains with a single variant difference, DBVPG6044 had four runs with two fewer variants called in comparison to the other runs and DBVPG1106 had one run with a single variant difference and three runs with two fewer variants called.

Strains	**True No. of Variants**	**Runs**
DBVPG1373	4714	1000
DBVPG6044	3104	996
	3102	4
L1374	2851	1000
SK1	2961	999
	2960	1
UWOPS83-787.3	2041	1000
W303	424	1000
Y55	2091	1000
YJM978	3981	1000
YPS128	2584	1000
BC187	580	1000
DBVPG1106	4649	996
	4648	1
	4647	3
DBVPG1788	3623	1000
DBVPG6765	3686	1000
L1528	2977	1
	2976	999
UWOPS03-461.4	3175	3
	3174	997
UWOPS87-2421	2276	1000
Y12	2841	1000
YJM975	3940	1000
YJM981	4012	1000

Table 3.6: **Deterministic Testing of the *vg* Variant Caller.** The number of variants called against the *vg* v1.26 reference graph on Set 1 of the simulated Bergstrom strains across 1,000 repeated runs.

3.3.8 Comparison of the *RTG vcfeval* Output

Analysis of the *vcfeval* comparison of the ground-truth VCF file generated through alignment of the simulated datasets and the S288c chromosome I reference sequence using *nucmer* for the various graph genome software identified several errors in the resulting variant calls. Many positions of the insertion and deletion events called from *nucmer* for the truthset were identified to be shifted one position to the left in comparison to the positions identified by the graph genome software for the same mutation event. This resulted in *vcfeval* evaluating the indels from the truthset as false negative variants whilst the same indels from the different graph genome software were evaluated as false positive variants. In addition, starting at chromosome I position 11,638 in the truthset, there were many spurious false positive SNP calls spanning the rest of the genome. These SNPs were not present in the FASTA sequence file for that genome, nor had any read support, as shown in Figure 3.11. Therefore, these SNPs could not be due to errors induced when simulating the sequence reads with ART. The false positive SNPs were present in the truthset of all of the strains examined in the first five datasets but were not present in the VCF files from any of the four graph genome software. Consequently, they were classified as false negative variants by *vcfeval*. As the false positive SNPs were present in the SNP files from the *nucmer* alignments, the SNPs were speculated to have arisen from incorrect local alignment of the simulated genomes in *nucmer*.



(a) DBVPG1373 Sequence Read Alignment Against S288c_ChrI

(b) Truthset VCF File

Figure 3.11: **Visualisation of Variants from DBVPG1373.** This figure displays a) the alignment of Set 1 DBVPG1373 sequence reads against the S288c_ChrI reference sequence using *samtools tview* and b) the truthset VCF file generated from *nucmer* containing the false positive SNPs spanning that region of alignment.

Further manual inspection of the true positive, false positive and false negative variant classifications by *vcfeval* also identified several variants predicted by all of the graph genome software that had been incorrectly classified. Out of the 112 false positive variants identified by the *vg* 1.26 reference graph for DBVPG1373, 13 of the variants were true positive variants that were represented in the exact same manner as in the truthset. These variants had the same reference allele, alternate allele and position. However, *vcfeval* was unable to evaluate them accurately, as shown in Figure 3.12. Three of the 12 false positive variants from the *Graph Genome Pipeline* and four of the 10 false positive variants from *Graphtyper* were also true positives with the exact representation that had been falsely evaluated by *vcfeval*. As the variants in the called VCF files were evaluated as false positives, the corresponding variants in the truthset were determined as false negatives.

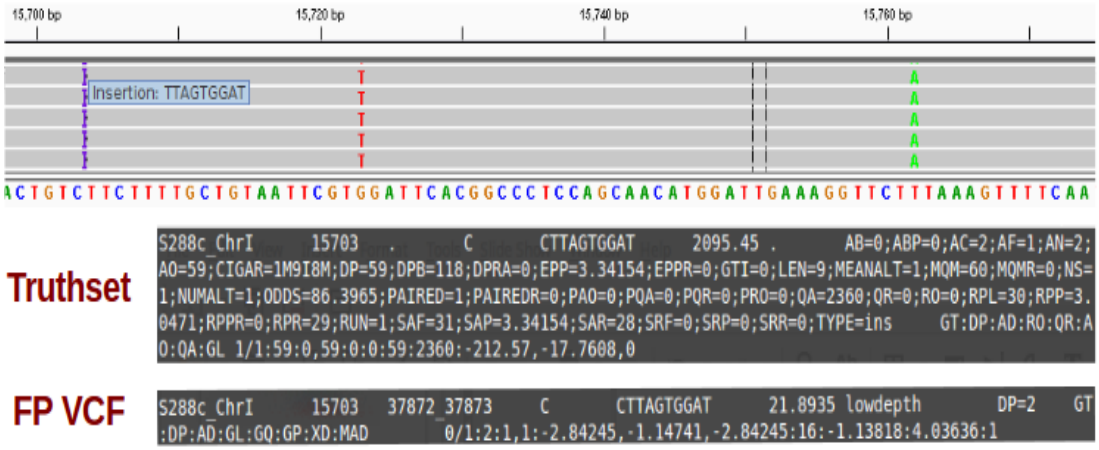


Figure 3.12: **Misclassification of True Positive Variants.** This figure shows the alignment of sequence reads against the *vg* 1.26 DBVPG1373 reference graph in the *IGV viewer*, highlighting the insertion at position 15,703. The TTAGTGGAT insertion sequence is shown to be present in the truthset VCF file but has been assigned to the false positive VCF file by *vcfeval*.

The variant calls in the second version of the truthset VCF file (i.e. generated through mapping of *pIRS* reads using *BWA* and subsequent *FreeBayes* variant calling) consisted of more complex variants, than in the previous analysis, with variants that occurred in close proximity to one another tending to be compounded into a single variant call. When comparing the compounded calls in the truthset VCF file against the graph genome called VCF files, which contained the same complex variant decomposed into multiple, smaller variants, *vcfeval* was unable to classify the variants in the called VCFs as true positives due to the differences in representation, as shown in Figure 3.13. The remaining six of the 10 variants in the *Graphtyper* FP VCF file were found to be proximate calls that were called individually in *Graphtyper* but combined in the truthset VCF file, therefore all 10 variants in the FP file for DBVPG1373 in the first dataset were evaluated incorrectly. Two of the 12 variants in the *Graph Genome Pipeline* FP VCF file were also inaccurately evaluated proximal variants.

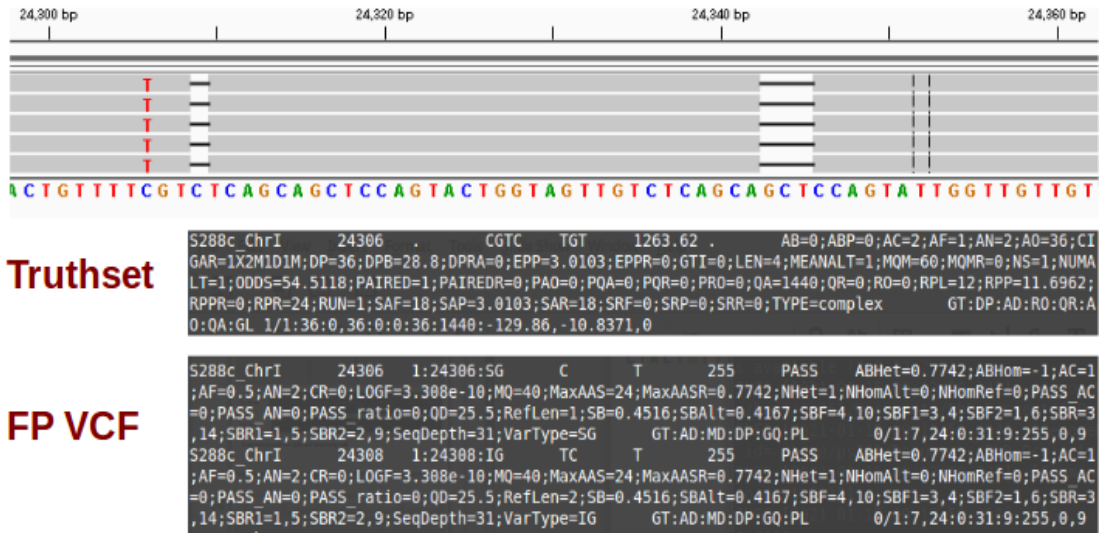


Figure 3.13: **Differences in the Representation of Proximal Variants.** This figure displays the alignment of sequence reads against the *Graphtyper* DBVPG1373 reference graph. The alignment features two proximal variants, a C -> T substitution at position 24306 followed by a deletion of the base C at position 24,309. Both variants have been compounded into a single complex call by *FreeBayes* within the truthset whereas they have been called individually in *Graphtyper*, resulting in *vcfeval* determining the variants to be false positives.

As *vg* is the only haplotypecaller amongst the graph genome software and due to the increased number of false positive variant calls made by *vg* found during the manual inspection of alignments, this resulted in haplotype calls in which there was both one correct and one incorrect call. The presence of the incorrect haplotype meant that the variant call was classified as a false positive by *vcfeval*, as shown in Figure 3.14. The *vg* 1.26 reference graph FP VCF file contained five haplotype calls where there was one right and one wrong call.

Following the generation of the second truthset VCF file, two problems remained in comparing truthset and predicted VCF files. Firstly, haplotype variant calls generated by both *FreeBayes* and *vg* needed to be decomposed. Secondly, errors in the *vcfeval* software when comparing variants needed to be resolved. In order to mitigate the first problem, it was necessary to normalise and decompose all variants into their most primitive form, using the *vcflib* post-processing pipeline, which enabled the recognition of identical variants by *vcfeval*.

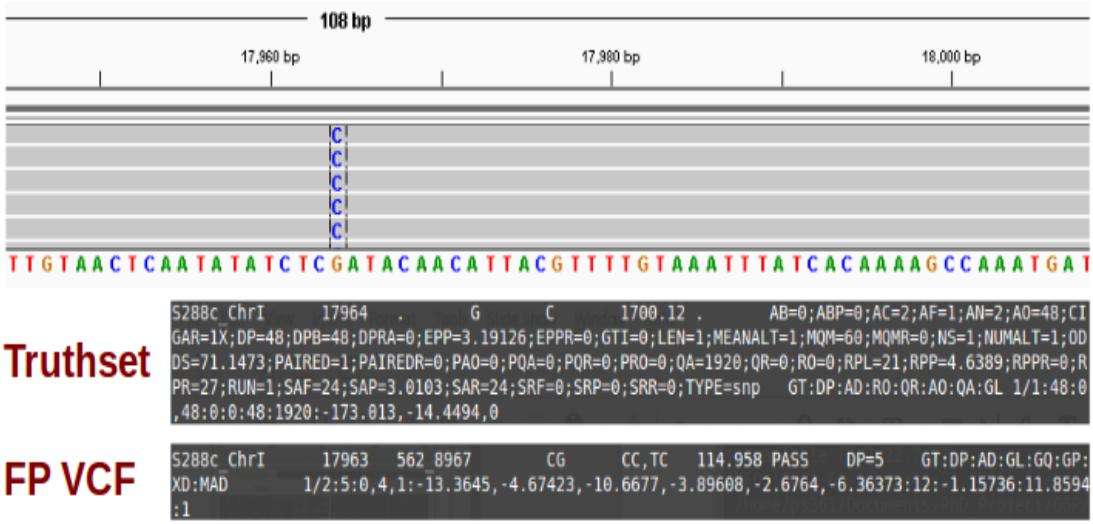


Figure 3.14: **Incorrect Haplotype Calls in *vg*.** This figure depicts the alignment of sequence reads against the *vg* 1.26 DBVPG1373 reference graph. The first haplotype call is the true positive variant whereas the second haplotype is a false positive due to the incorrect C -> T substitution called at position 17,963. This resulted in the variant call being classified as false positive by *vcfeval*.

3.3.9 Comparison of the Simulated Dataset Variant Calling Across Different Graph Genome Software

The corrected *vcfeval* classification was used to compare the accuracy of the filtered variant calls from all 12 references for each of the 19,000 simulated chromosomal datasets. The number of true positive variant calls was compared against the number of false positive variant calls for each dataset, as shown in Figure 3.15. The individual peaks seen on the graph can be attributed to the different numbers of variants within each strain. *BayesTyper* was found to call the least number of false positive variants. However, for several genomes, it was also found to call fewer true positives than was expected for that strain, demonstrated by the outlier values in the bottom left of the scatter plot. The *vg* 1.26 variation graph was shown to call the highest number of false positives across all of the datasets compared to the other references, followed by the *vg* 1.5 variation and reference graphs. For both *vg* 1.26 and *vg* 1.5, the variation graph calls contained more false positives than the reference graph calls. The *vg* 1.18 and *vg* 1.5 variant calls were also shown to contain fewer true positive variants in comparison to the other software (see Appendix Figure B.19 for the scatter plot without *BayesTyper*). Strains that contained the least number of variants, such as W303 and BC187, had a lower number of false positive variant calls with the least variance in the range of the number of true positive variants called by each software in comparison to more highly divergent strains.

Comparison of the true positive variant calls against the false negative variants has been displayed in Figure 3.16. As expected, the scatter plot clearly shows that as the number of true positive variants called decreases with each software, the number of corresponding false negative variants was also shown to increase. *BayesTyper* was shown to have the lowest number of false negative variants for the majority of the datasets compared to the other variant callers. However, it also called the highest number of false negative variants for several of the datasets as depicted by the outlier values in the top left of the scatter plot. For all of the graph genome software, variant calls made from the variation graph were found to contain fewer false negatives in comparison to the reference graph variant calls (see Appendix Figure B.20 for the same scatterplot without *BayesTyper*). The *Graph Genome* toolkit also had the lowest number of false negative variant calls, followed by *BWA*, *Graphtyper*, *vg* 1.26, *vg* 1.18 and *vg* 1.5, the latter of which predicted the most false negative variants.

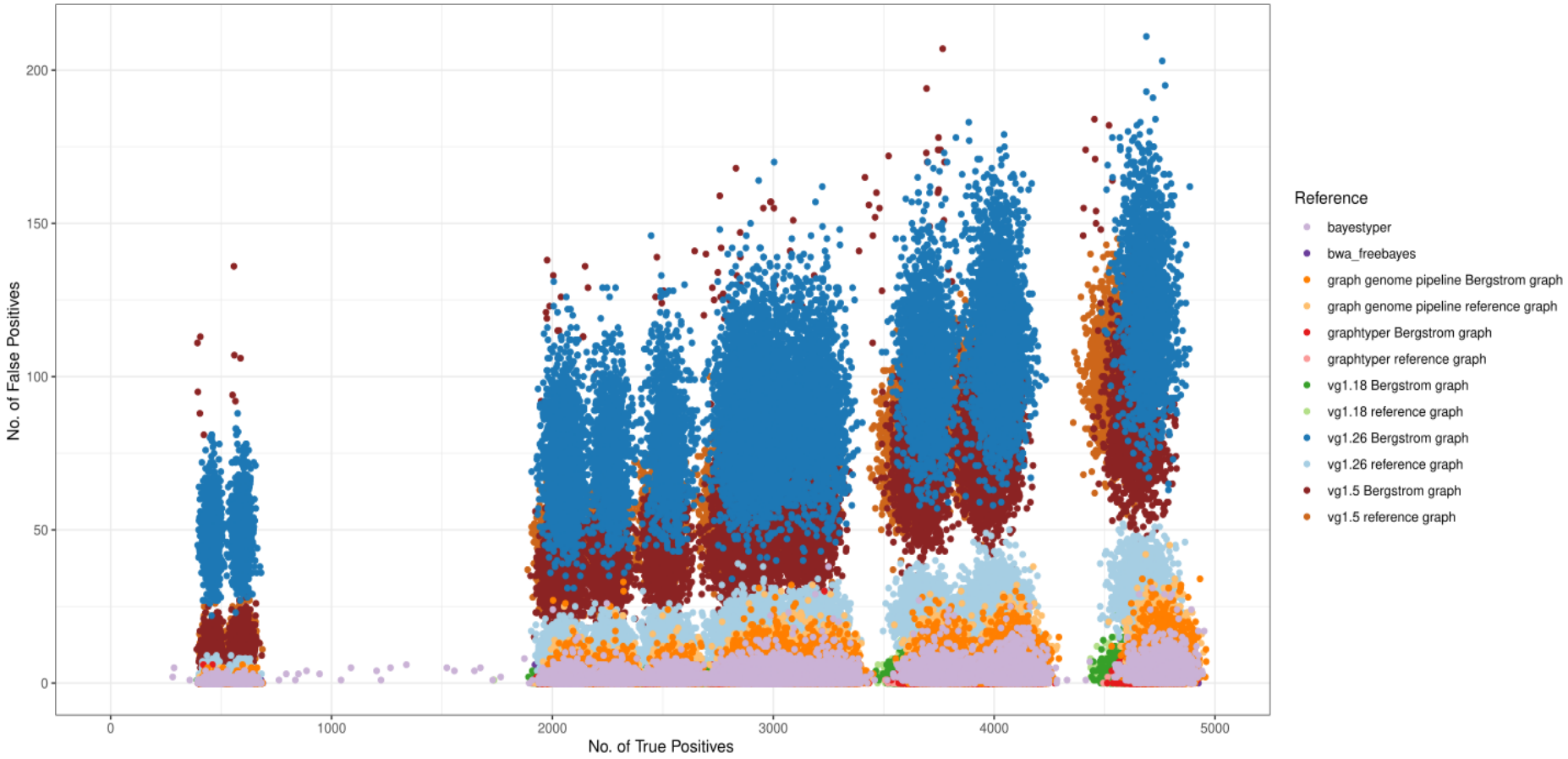


Figure 3.15: **True Positive vs False Positive Variant Calls.** The scatter plot depicts the no. of true positive and false positive variant calls from 12 references across 19,000 datasets.

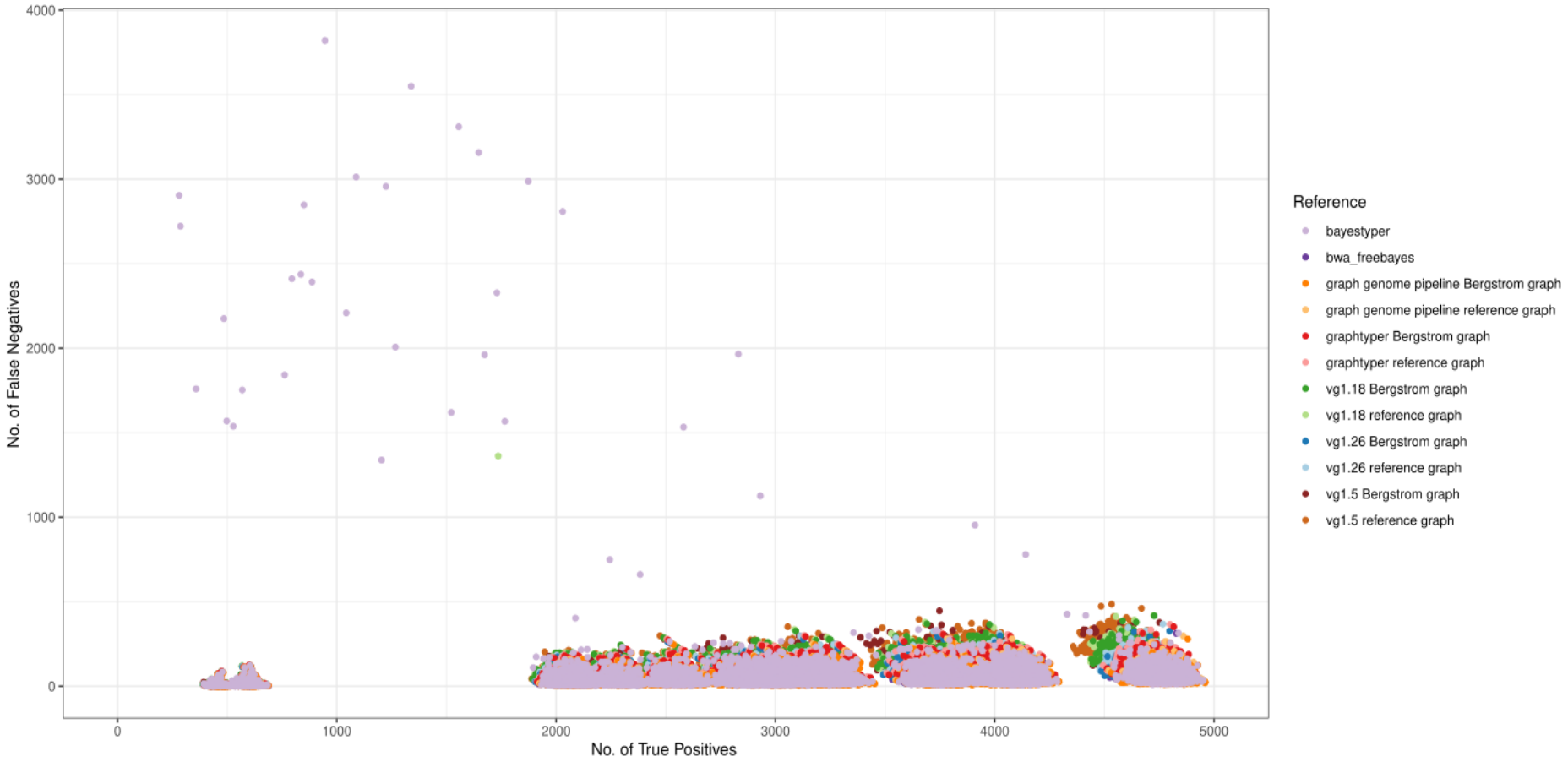


Figure 3.16: **True Positive vs False Negative Variant Calls.** The scatter plot depicts the no. of true positive and false negative variant calls made from 12 references across 19,000 datasets.

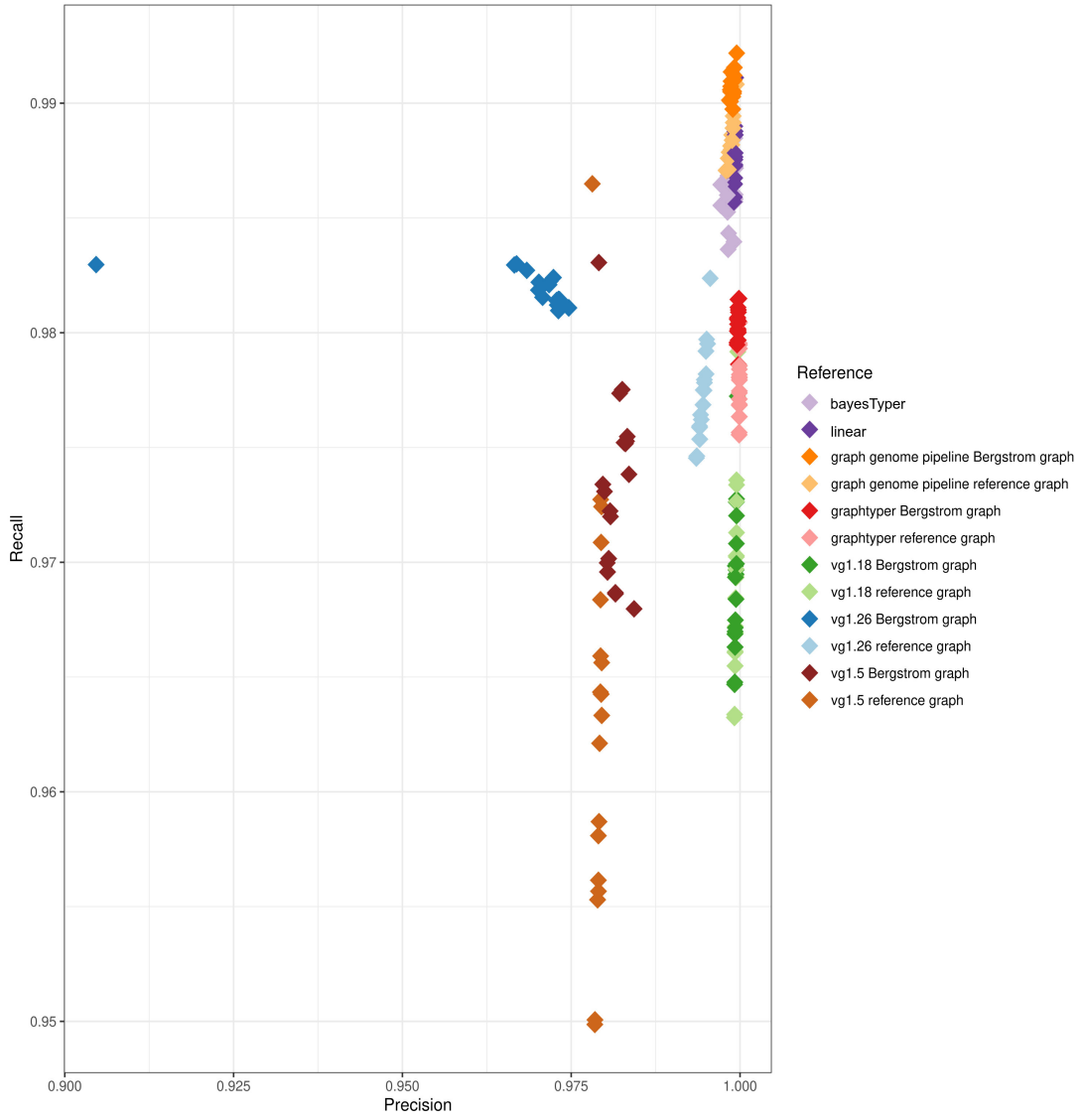


Figure 3.17: **Precision vs Recall.** This plot shows precision against recall values averaged over 1,000 datasets, for each of the 19 strains, for the 12 different references.

The accuracy of the variant calling algorithms was determined by comparing the level of precision against recall by taking the average of both values for all datasets and for each of the 19 strains, as shown in Figure 3.17. All of the variation graphs were found to have better recall, which accounts for the false negative variants, in comparison to the reference graphs, except from *vg* 1.18 for which there was no difference between the two. The *Graph Genome Pipeline* variation graph had the highest score for recall, followed by the *Graph Genome Pipeline* reference graph. This is further supported by Table 3.7 which shows the average values across the 19,000 genomes for each reference structure, in which the overall recall score for the *Graph Genome Pipeline* variation and reference graphs are 0.991 and 0.988, respectively. The differences in the average

number of true variants (the total number of true positives and false negatives) predicted was found to vary for each software due to the differences in variant representation, as discussed in Section 3.3.8. The variants called from the linear reference genome by *FreeBayes* were shown to have better recall than *Graphtyper* and *vg*, with the average precision and recall over all datasets being the same as for the *Graph Genome Pipeline* reference graph. *BayesTyper* was also found to have high recall with a slightly lower precision, averaging 0.998 and 0.986 across all of the datasets. The *vg* 1.5 reference graph was found to have the lowest recall in variant calling, averaging 0.964 across all of the genomes. The level of precision in variant calling was also found to very high amongst all of the software with there being little to no difference between the variation and reference graphs, apart from *vg* 1.5 and *vg* 1.26. The *vg* 1.26 variation graph was found to have the lowest precision score for variant calling, with the precision for the mosaic laboratory strain W303 dropping far below the other strains. *Graphtyper* was found to have perfect precision scores (1) for both the reference and variation graphs.

Reference	**True Positives**	**False Positives**	**False Negatives**	**Precision**	**Recall**	**F1 score**
GGP var	3037	4	29	0.999	0.991	0.995
GGP ref	3028	5	37	0.999	0.988	0.993
Linear ref	3015	2	40	0.999	0.988	0.993
BayesTyper	3022	2	41	0.998	0.986	0.992
Graphtyper var	3008	1	61	1.000	0.980	0.990
Graphtyper ref	2999	0	69	1.000	0.978	0.989
vg 1.26 ref	2982	17	71	0.995	0.978	0.986
vg 1.18 var	2960	2	97	0.999	0.970	0.984
vg 1.18 ref	2959	2	98	0.999	0.970	0.984
vg 1.5 var	2963	56	85	0.982	0.974	0.978
vg 1.26 var	2990	89	56	0.965	0.982	0.974
vg 1.5 ref	2931	63	120	0.979	0.964	0.972

Table 3.7: **Comparison of the Variant Calls Average Across 1,000 Datasets.** The average values of the number of variants and the precision, recall and F1 scores calculated from the 19,000 genomes. The ref in the Reference column stands for reference graph and the var stands for variation graph.

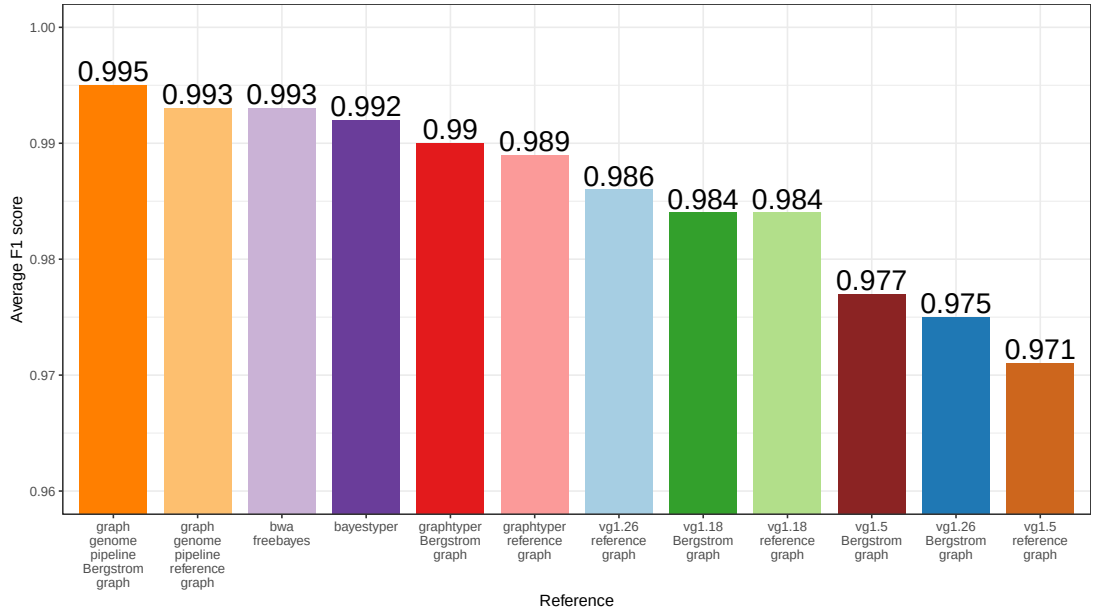


Figure 3.18: **Average F1 Scores.** This bar chart displays the average F1 score calculated from all of the datasets for each reference structure, ordered by their F1 scores.

The average F1 score, which account for the trade-off between precision and recall, from all of the datasets, were compared to determine which software and reference structure produced the most accurate variant calls overall. The bar chart in Figure 3.18 shows that the *Graph Genome Pipeline* variation graph had the highest F1 score, 0.995, followed by the *Graph Genome Pipeline* reference graph and the linear reference at 0.993. *BayesTyper* was found to perform better than the *Graphtyper* variation graph as it had slightly better recall. Due to their increased ability to recall variants, the variation graphs had greater F1 scores in comparison to the matching reference graphs, with the exception of *vg* 1.26 in which the reference graph outperformed the variation graph as a result of its reduced precision. The *vg* variant calling algorithm was found to produce the least accurate variant calls of the graph genome software, with *vg* 1.5 performing worst overall.

3.4 Discussion

The analysis of sequence read alignments from different mappers has highlighted the need for a single metric that could be utilised as a standardised method to assess the accuracy of mapping. There are already a few measures widely in use for characterising

alignments, such as the CIGAR string, the MD tag and the CS tag. However, each of these are measures are limited in the alignment information they present. Moreover, there is a need for a measure that captures an exact base by base representation of how sequence reads map to a reference structure yet there is no known script or software available that could provide this information. The *FAT-CIGAR* toolkit implemented here allows for a true read alignment to be obtained, whether for a linear or a graphical reference, which can then be used to determine the accuracy of a mapping algorithm.

The run time for the *FAT-CIGAR* toolkit on linear alignments was shown to be vastly quicker than for the *vg* reference graph. For *BWA* alignments, the FAT-CIGAR string is constructed on the fly from the information within the BAM file, therefore the file does not need to be loaded into memory. The *vg* alignments have a much longer run time as they require large JSON files to be loaded into memory and indexed using hash tables before each read is cross-searched from the BAM file. Even with the hash optimisation for read searching, an increased number of sequence reads can result in JSON files that are several hundred gigabytes in size. The overall run time of the *FAT-CIGAR* toolkit could be greatly reduced for *vg* alignments by breaking up the GAM file into multiple smaller files for each chromosome using the *vg chunk* function, converting the chromosome GAM files into JSON and BAM files and running the script on each individual chunk for strains with over five million reads. This could be incorporated as a function in the toolkit, where the file size is checked prior to calling *vg* from within the script to split large GAM files before running. The per-chromosome files could then be processed in parallel before combining to output a single, sorted BAM file. However, the *vg* software is currently still under rapid development with major changes being made constantly to the core algorithm that introduce redundancy to certain functions. Therefore, it would be most efficient to add such a feature to the *FAT-CIGAR* toolkit once the *vg* software has stabilised.

The use of a variation graph reference structure was clearly shown to increase the number of sequence reads that are able to align against the reference for both *vg* and the *Graph Genome* toolkit. This was expected as incorporation of the variants within the population in the reference structure facilitates read alignment, particularly within the highly polymorphic regions of the genome. Interestingly, it was noted that even in the absence of variants, a greater proportion of sequence reads were able to align against the

graph-based reference structure in comparison to the standard, linear reference. These findings have also been supported in the studies published on both *vg* and the *Graph Genome* toolkit (Garrison *et al.*, 2018, Rakocevic *et al.*, 2018). This observation may be due to the nature of aligning against a graph-based reference structure requiring these mapping algorithms to have increased alignment sensitivity. The increase in mapping against the *vg* reference graph compared to the *Graph Genome* toolkit variation graph demonstrates that the *vg* mapping algorithm has the greatest sensitivity of the alignments studied at aligning sequence reads.

The sequence identity score calculated from the FAT-CIGAR string was able to provide an accurate depiction of alignment quality. The reduction in the percentage of reads with perfect mapping when using the true sequence identity scores was due to the reassignment of some matches in the CIGAR string to mismatches in the corresponding FAT-CIGAR string. As the CIGAR string is routinely used to calculate the sequence identity for linear read alignments, this lack of distinction previously enabled alignments that contained several mismatched bases to be considered as perfectly mapped. In addition to increasing the quantity of mapped reads, the use of a variation graph reference structure was also shown to vastly improve the quality and accuracy of alignments by increasing the percentage of perfectly mapped reads. As the variant alleles are already present within the reference, this reduces ambiguity in mapping and eliminates reference bias resulting in improved alignment accuracy. The *Graph Genome* toolkit variation graph was found to align the greatest proportion of reads with sequence identity scores greater than 0.9 suggesting that the alignment accuracy was better than that of the *vg* mapping algorithm. The overall proportion of well-mapped reads were also found to be slightly higher in *BWA* compared to both the *vg* and *Graph Genome* toolkit reference graphs. This suggested that despite the increase in the quantity of mapped reads against the reference graphs, in the absence of variants the quality of alignment remains the same.

The *FAT-CIGAR* toolkit also enabled direct comparison of alignments against each of the different reference structures. The reduction in the number of reads that aligned with the same CIGAR string to the same FAT-CIGAR string was mainly due to the reads that aligned against the variation graphs in *vg* and the *Graph Genome* toolkit. The absence of variants within the reference structure meant that read alignments

against both the reference graphs and linear reference genome are more likely than the variation graph to contain mismatched bases due to the presence of SNPs, which these mismatches highlighted by the FAT-CIGAR string but not the corresponding CIGAR string. The presence of alternate alleles within the variation graph enables the same reads to align as perfect matches, leading to the deserved increase in reads that map at the same position yet with a different FAT-CIGAR string. Furthermore, use of a graphical reference structure also led to an increase in reads that aligned at different positions as the likelihood of mapping to the true location is increased when aligning against the variation graph. As the majority of the differences in mapping and position arose from read alignments against the variation graph, this further supported the idea that mapping against a graph-based reference improves the quality of alignment.

Unsurprisingly, comparison of the number of variants called across each of the four graph genome software showed that an increased number of both SNPs and indels were called against the variation graphs in comparison to the reference graph. This suggested that the increase in sequence read mapping against the variation graph and the elimination of reference bias in mapping translates to increased sensitivity in the variant calling algorithms enabling novel variant discovery. As with mapping, the wide range seen in the number of variants called across the Bergstrom strain dataset can be mainly attributed to a very high variance in the quality of the sequencing reads, as seen previously in Chapter 2. The heightened difference in the number of variants called between the *vg* variation graph and the other variation graphs also indicated the possibility of an increase in false positive variant calls. However, the potential presence of false positive variants within the variant calls could not be examined due to the absence of a curated ground truthset for *S. cerevisiae*, such as the Illumina Platinum Genomes (Eberie *et al.*, 2017) and Genome in a Bottle Consortium (Zook *et al.*, 2014) bench-marking datasets for human genomes.

Generation of the 1,000 Bergstrom-like simulated datasets using *TreetoReads* was an extremely lengthy process, as it required several other sub-processes and software to be run in order to obtain accurate input parameters. The branch lengths of the input tree are scaled by *TreetoReads* which resulted in far too many variants being introduced into the more distant strains due to the greater branch lengths. In order to achieve the required numbers of variants across all strains, the *TreetoReads* Python code had to be

edited to scale down the branches of the distant strains based on the branch length, as confirmed with the author of the software, Emily McTavish. The *TreetoReads* algorithm uses a non-deterministic approach so testing procedures to find the optimal parameters resulted in it being run 172 times prior to generating the final simulated datasets.

The *BayesTyper* variant calling algorithm was found to be non-deterministic, resulting in unstable variant calls that change with each run. Therefore, determining the genotype of a strain from a single run would not be highly accurate. Due to the limitation of time constraints for this project, each dataset could only be run once as it took several weeks to complete the analysis of all 1,000 datasets. Both for this work, in order to allow for fairer comparison, and for any future users of the *BayesTyper* software, it would be advisable to take a consensus approach in which the variant calls across multiple runs are consolidated to obtain more accurate variant calls. Additional deterministic tests also had to be run on the *vg* algorithm to confirm stability as initial faulty runs resulted in the under-calling of variants for many datasets. This was found to be due to *vg* running out of memory space as the variant calling process is run in multiple steps that require large amounts of computational memory to generate an augmented graph containing the variants. Multiple datasets were initially run in parallel in order to expedite the run time, as they take several weeks to complete, which resulted in run time errors that produce the faulty variant calls. The final runs were carried out serially, confirming that the algorithm was not deterministic.

Prior to final variant call comparison analysis, the filtered variant calls from all of the software for the first DBVPG1106 dataset were manually curated again after variant classification in order to ensure there were no remaining software errors other than the ones identified and corrected regarding truth-set generation and vcfeval-based comparison. Generation of a yeast variant truthset enabled the accuracy of graph genome-based variant calling to be tested on a dataset other than the human genome, as this was the only species that had been used by the software developers to study the performance of the graph genome software prior to this study, with the sole exception of *vg*. RTG's *vcfeval* was utilised as it is the only software currently available that can carry out the required variant classification. For example, the only other software for benchmarking variant call sets, Illumina's hap.py, also utilises *vcfeval* to carry out variant comparison (Krusche *et al.*, 2019). *Vcfeval* was found to perform well overall in

identifying identical variants from different call sets but it also failed for certain variants when comparing a normalised call set against a VCF file that had not been normalised. Furthermore, it was surprisingly shown to fail for some identical variants, the reason for which is still unclear. It was important that these errors were corrected as it would have lowered the precision and recall for all of the software, making the variant calls seem less accurate. The preliminary analysis of the variant calls showed that the *vg* variant calling algorithm was unstable as it had called a significantly greater number of false positive variants in comparison to the other software. This was initially suspected to be due to a software version issue as the study examining read mapping identified that the *vg* mapper performs well at read mapping. Both *vg* 1.18 and *vg* 1.5 were subsequently run in order to test for version stability, as they had been used to carry out the work in the previous Chapter.

All three versions of *vg* were found to have the least accurate variant calls in comparison to the other graph genome software or *FreeBayes* for the linear reference. Depending on version, this was due either to calling a greater number of false positive and false negative variants or calling fewer true positive variants. As this lower performance was consistent across all three versions of *vg*, it strongly suggested that there is an inherent issue with the *vg* variant calling algorithm albeit with the mapping algorithm performing accurately. The *Graph Genome* toolkit was determined to have produced the most accurate variant calls as it had high precision due to the low number of false positives. Furthermore, it had the highest sensitivity of the variant calling algorithms, particularly for the variation graph. The *Graph Genome* toolkit is also quite computationally intensive and running the 1,000 datasets on my laptop with 8GB RAM took nearly two months to complete. *FreeBayes* was found to perform better than *BayesTyper*, *Graphtyper* and *vg*, which can be owed to the accuracy of the variant caller and the lack of complex structural variants within the simulated genome that would otherwise be present in a real, biological dataset in which the use of a variation graph would be highly beneficial. Even though there were many outlier variant call sets produced by *BayesTyper* due to the instability of the algorithm, as the majority of datasets were called with a high number of true positive variants and few false positives, this was averaged out across the 1,000 datasets to generate a high F1 score. The low number of false positive variants can also be attributed to its inability to call novel variants, as all of the variants genotyped were prior variants called using *FreeBayes*, *Platypus* and *Manta*

for each dataset. *BayesTyper*'s inability to call novel variants is a major limitation to its usability as the identification of these variants is highly important for studying and characterising new genomes. *Graphtyper* was found to contain the most precise variant calls, with little to no false positive variants across the majority of the genomes, particularly for the reference graph. *Graphtyper* was also the most user-friendly of the graph genome software and had the quickest run time as it processes multiple smaller regions in parallel, even allowing for the prior linear alignment that had to be carried out with *BWA*. Therefore, it would be the recommended software if avoiding false positive variant calls is of utmost importance in a study. Such a scenario might be, for example, when conducting a Genome-Wide Association Study.

3.5 Conclusion

Developing the pipeline for a large-scale simulation study which required the simulation and accurate genotyping of 19,000 genomes to be utilised as benchmarking datasets proved to be highly challenging, mainly due to the identification of many software errors that had to be corrected in order to carry out a fair comparison. The use of variation graph reference structures was shown to improve both the quantity and quality of sequence read alignment and increase the accuracy of variant calling. Variation graphs were shown to have much greater recall and better precision than their corresponding reference graphs supporting the theory that encoding alternate alleles within a reference structure can eliminate reference allele bias in mapping. The recommended software to obtain the most accurate variant calls was found to be the *Seven Bridges Graph Genome* toolkit. In functional genomic studies where it is important to minimise false positive variant calls, *Graphtyper* would be better suited owing to its near perfect precision. Although the work in the previous chapter had been carried out using *vg*, inherent issues identified with the *vg* variant calling means that it is not the most appropriate software for carrying forward with the analysis of yeast genomes. Graph genomics is still a relatively new field but the increase in the number of graph genome software released, due to the demand for alternate reference structures that can overcome the limitations of the linear reference genome, is a strong indicator that the use of variation graphs will eventually become the new normal for variant discovery.

The *FAT-CIGAR* toolkit, pivotal to obtaining these findings, permits the generation of true alignments for a range of different read mappers, alongside correct alignment scores for graph-based and linear read mappings, the latter of which *BWA* does not provide for all alignments. The FAT-CIGAR string also enables prior filtering of read alignments within the BAM files which can in theory improve the accuracy of variant calling and further downstream analyses. Improvements in variant calling made by read filtering will be discussed in greater detail in the following chapter.

Chapter 4

FAT-CIGAR : A Novel Method of Variant Filtration

4.1 Introduction

The exponential increase in next-generation sequencing data has been a driving motivation for the development of numerous software for read mapping and variant calling in a constant effort to improve the accuracy of current data analysis pipelines. The transition towards personalised genomics and medicine over the recent years has also meant that whole-genome sequencing and whole-exome sequencing are more widely used in clinical settings for the identification of causal variants in disease diagnostics. This has allowed for the discovery of rare variants in disease and for treatments to be specifically tailored towards each individual. Thus, there is an increasing requirement for variant prediction methods to be carried out with the utmost degree of precision.

The key challenge in variant prediction is to be able to distinguish between the true variants and false positive calls. Various approaches such as quality control analysis of the sequencing data, filtering sequence reads based on mapping quality and carrying out local realignment of large indel regions can contribute towards improved accuracy in variant calling. Post variant-calling, variant filtration methods can be utilised to identify and remove false positive variant calls. The main traditional approach is to carry out hard filtering on the variant calls using the annotations provided by the variant caller. These annotations, such as the read depth, genotype quality and minor allele frequency, can be found within the INFO and FORMAT fields of the VCF file. Variants can be filtered on one or more thresholds as specified by the user. A major

limitation of this approach is that the accuracy of filtering is dependent on choosing the correct filtering thresholds. The user is required to have knowledge of the dataset in order to select appropriate filtering thresholds and ensure that the filters are not overly stringent. Hard filtering may also be incapable of removing high confidence false positive variants that have strong read support.

The *Variant Quality Score Recalibration (VQSR)* tool by *GATK* (DePristo *et al.*, 2017) is a specific example of a dedicated variant filtration tool. *VQSR* utilises a machine learning algorithm to compare the variant calls against a highly validated truthset on which the algorithm is trained. The annotations from the truthset are utilised to identify appropriate filtering thresholds. New quality scores are assigned to the variants and used to filter out false positive variant calls. *VQSR* can provide greater accuracy in identifying true variants for species that have a highly curated call set, such as the HapMap project (The International HapMap Consortium, 2003) and 1000 Genomes project (1000 Genomes Project Consortium, 2015) for human genomes. However, in the absence of a curated call set, as is unfortunately the case for many species, this method of filtering cannot be utilised (GATK, 2021).

In this chapter, we aim to introduce a relatively new method of variation filtration that utilises the *FAT-CIGAR* toolkit to filter out reads that are not terminally anchored against the reference genome, by a specified number of bases. The effectiveness of this method of filtering at removing false positive variants prior to variant calling will be highlighted. The differences in the accuracy of variant calling will be observed when filtered on the CIGAR and FAT-CIGAR strings. In order to compare the accuracy of the variant calls, the *sim_genomes* program will be developed to combine the process of simulating genomes and generating the truthset for each genome. Several genomes will be simulated with varying numbers of SNPs and indels and the accuracy of filtering will be determined by the F1 scores to identify the most optimal read filter. Read filtering will also be carried out on *S. cerevisiae* strains that were re-sequenced at different times and using different technologies. The sequence reads from each strain will be compared against itself to examine the changes in the proportion of shared variants.

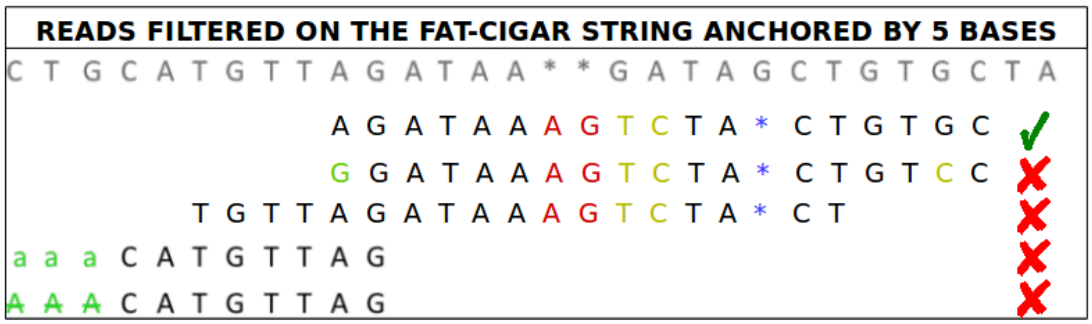
4.2 Methods

4.2.1 The *FAT-CIGAR* Toolkit: Read Filtering

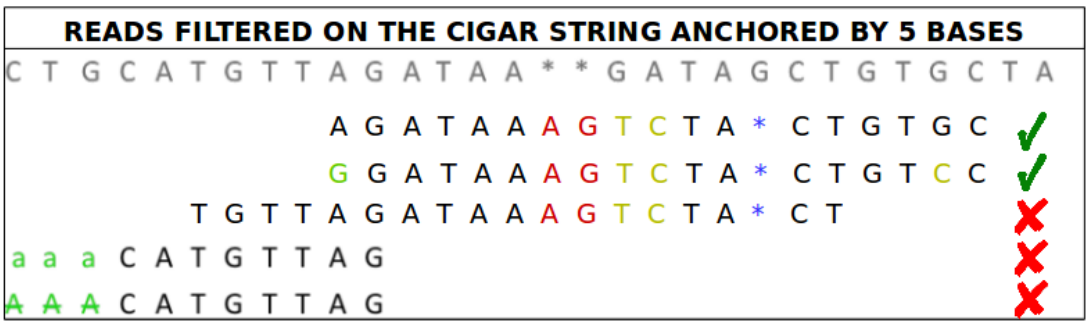
A read filtering option was added to the *FAT-CIGAR* toolkit, which was introduced in Chapter 3, to enable the user to specify a certain number of bases by which the read has to be anchored against the reference sequence by exact matches at either ends (see Figure 4.1). The idea for this method of read filtration was inspired by techniques utilised in the *TURNIP* (Davey *et al.,* 2010) and *Parsley* (https://github.com/ziaursani/parsley_-root) software by Dr. Jo Dicks and Dr. Robert Davey. This method of anchoring reads to the reference using the *FAT-CIGAR* toolkit was tested as a relatively novel variant filtering technique to prevent false positive SNP calls that arise from misalignment at read ends. As the *FAT-CIGAR* toolkit permits for both the non-surjected CIGAR and FAT-CIGAR strings to be obtained for alignments from various mapping software, either string could be utilised by the user for read filtering. The *-a/–fat_ cigar_ anchor* option should be specified to filter on the FAT-CIGAR string or the *-ca/–cigar_ anchor* option should be specified to filter on the CIGAR string. If either read filtering options are specified, for each mapped read the chosen string is checked to ensure that both the first and last operations in the string represent base matches. If not, the read is immediately discarded from the BAM file. If there are matches at either ends of the read, the number of bases is compared and only reads with base matches greater than or equal to the number of bases specified by the user are retained and written to the output BAM file.

4.2.2 The Effect of Read Anchoring Length on Read Retention

In order to test whether read filtering with the *FAT-CIGAR* toolkit can improve variant calling, it was important to identify the optimal base length for read anchoring. The *pIRS simulate* function was run with *-x 30* (read coverage) and default parameters to simulate paired-end Illumina sequence reads from chromosome I of the S288c reference genome. The simulated reads were aligned against the S288c FASTA file using *bwa mem*. The resulting alignment SAM file was converted to BAM format, sorted and indexed using *SAMtools*. *Picard's MarkDuplicates* function was used to remove PCR duplicates from the BAM file prior to read filtering. The *FAT-CIGAR linear* function was run with the *-a* parameter to anchor reads by filtering on the FAT-CIGAR string.



(a) Filtering on the FAT-CIGAR String



(b) Filtering on the CIGAR String

Figure 4.1: **Read Filtering using the *FAT-CIGAR* Software.** This figure is an example of sequence reads that were filtered by anchoring the read ends against the reference sequence by five bases using the FAT-CIGAR software. The green tick represents reads that passed the filter whilst the red cross represents reads that were removed during filtering. Figure 4.1a shows sequence reads that were filtered on the FAT-CIGAR string. Only the first read was able to pass the filter as both read ends contained exact base matches against the reference. Figure 4.1b shows sequence reads filtered on the CIGAR string. The second read was also able to pass as the 'G' and 'C' SNPs are masked as base matches in alignment by the CIGAR string. This figure was adapted from Dündar *et al.*, 2015.

Read anchoring was carried out with 5, 10, 15, 20, 25 and 30 bases and the retention of reads within the BAM file was monitored.

4.2.3 Studying the Impact of Read Anchoring on Variant Calling

In order to assess whether read filtering can improve variant calling, it was critical to generate a truthset against which the called variants can be evaluated. Simulated truthsets were utilised to test the hypothesis that filtering sequence reads on the FAT-CIGAR

string can reduce the number of false positive variants called, allowing for greater accuracy in variant calling. The *sim_ genomes* Python script (https://github.com/prithikasritharan/sim_genomes) was written to simulate genomes from a reference sequence and record the exact variant alleles within the simulated genome. As before, S288c chromosome I was chosen as the reference from which to simulate the genomes. Initially, only SNPs were simulated within the genomes to focus on how read anchoring affected SNP calls. The *sim_ genomes* Python script simulates SNPs under the assumption that substitution events are independent and occur at random within the genome. It takes in the number of SNPs to simulate, the input FASTA file and a prefix name for both the output FASTA and VCF files as required input arguments. The reference sequence from the FASTA file is stored into memory and the *random* Python library is used to determine the positions at which to insert the SNPs, ascertaining each position is only selected a maximum of once. At each chosen position, the exact SNP is also decided using the *random* library and each variant allele is checked against the reference allele to ensure it is not the same. If the randomly generated variant allele is the same, the reference allele is mutated again until a SNP is produced. The simulated genome is written out to the user specified FASTA file with the header stating the number of variants within the sequence. The benchmark VCF file is also written out containing the following information: the position, reference allele, variant allele, pass filter, variant type and the genotype. Three genomes were simulated containing 1,000, 2,000, and 3,000 SNPs.

Prior to simulating sequence reads, the *pIRS* in-house error profiling scripts were utilised to generate error profiles specific to *S. cerevisiae*, following the methods outlined in the *pIRS* documentation. As NCYC78 was previously found in Chapter 2 to have high sequence read quality, it was selected for profiling. The substitution error profile was generated using a two-step alignment process in which *bwa aln* was used to map the NCYC78 forward and reverse reads FASTQ file against the reference separately, producing an SAI intermediate file containing the suffix array indexes. The SAI files were provided as input with the FASTQ files to *bwa sampe* to obtain the final alignment SAM files. The *pIRS baseCalling_ Matrix_ calculator* script was run with the SAM file and the S288c_ChrI FASTA reference file to generate the substitution error profiles. The GC error profile was obtained using the *SOAP2 v2.21* (Li *et al.*, 2009) aligner. The *2bwt-builder* was used to index the reference and *soap* to align the reads with the

parameters: *-s 40* (minimal alignment length) and *-l 32* (seed length). The average read coverage was estimated using *soap.coverage* and the output depth files were run with the *pIRS gc_coverage_bias* script to obtain the GC error profile. The indel error profile could not be generated as the *pIRS indelstat_sam_bam* perl script produced a read length error when run with the NCYC78 SAM file. It was not possible to debug this error as when the script was edited to bypass the error code, the script ran but did not produce any output therefore, the standard indel error profile was utilised instead. *pIRS* was used to simulate sequence reads from each of the three simulated genomes using both the default *pIRS* error profile and the NCYC78 error profile to identify any errors in read simulation. The reads were mapped using *bwa mem*, sorted, indexed and PCR duplicates were removed. The *FAT-CIGAR* toolkit was used to anchor reads on the FAT-CIGAR string by 10 and 20 bases. Variants were called from the unfiltered and filtered BAM files using *FreeBayes* and the *vcflib* post-processing pipeline (see Chapter 3) was utilised to normalise and decompose the variants. The accuracy of the variant calls were compared against the truthset using *RTG's vcfeval* to classify the variants. The variant classifications were further corrected using the Python script developed in Chapter 3 to ensure any misclassified false positive and false negative variants were adjusted appropriately.

The *sim_genomes* Python script was further expanded to allow for the simulation of indels in order to test whether the *FAT-CIGAR* toolkit can reduce false positive indel calls. The number of indels was added as an additional required input argument. The choice of different types and proportions of indels to simulate was adapted from the method of indel simulation utilised in the *SInC* (Pattnaik *et al.*, 2014) software (see Figure 4.2). The number of each indel type was determined as a percentage of the overall number of indels specified by the user. Single base indels were simulated for 35% of the total variants, repeat regions were expanded for 10% of the variants and 55% of the variants were non-repeat indels which consisted of 96% medium indels, varying in proportions ranging from 2 to 20 bp and 4% long indels between 20 - 100 bp. The *TRF (Tandem Repeat Finder)* v4.09.1 (Benson, 1999) software was run using the recommended parameters on the S288c_ChrI reference sequence to identify the repetitive regions of the genome which were used to carry out the repeat expansions. The DAT file output from TRF contained 46 possible repetitive regions and if there were different possible repeat sequences from overlapping regions, the sequence with the

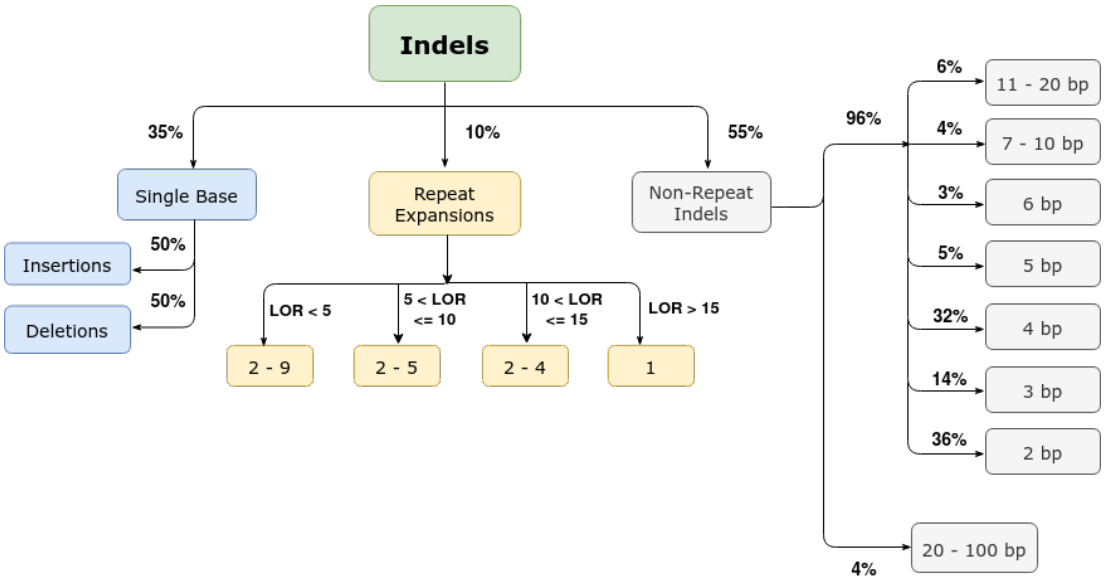


Figure 4.2: **Simulating Indels.** This flowchart shows the different types and percentage of indels that were introduced within each simulated genome. The three types of indels were single base (blue), repeat expansions (orange) and non-repeat medium and long indels (grey). For the repeat expansions, LOR refers to the length of the repeat sequence with the orange boxes showing the number of potential repeats.

greatest alignment score was chosen, leaving 38 repeat sequences in the final output. The script read in the DAT file containing the starting position, end position, the length of the sequence and the repeat sequence into a list. For each expansion, the repeat sequence was chosen at random and the number of expansions was also chosen at random, based on the length of the repeat sequence, as detailed in Figure 4.2. The end position of the selected repeat sequence was determined as the starting position from which to insert the expansions. For the other indel types, the position of the indel was generated at random ensuring that each position was unique. The length of the indel sequence was also chosen at random for the non-repeat medium and large indels ranging from 7-10 bp, 11-20 bp and 20-100 bp. As with the single base indels, 50% of each type of non-repeat indels were insertions and 50% were deletions. The insertion sequences were also generated using the *random* Python library. As the introduction of each variant allele changed the positions within the simulated genome, a list of positions in regard to the reference was maintained to identify the exact position in the simulated genome. The final simulated genome containing both SNPs and indels was written out to a FASTA file whilst the variants were written to a VCF file with the variant types defined as follows: "complex" for repeat expansions, "ins" for insertions and "del" for

deletions.

Three different genomes were simulated each containing 1,000 SNPs and either 100, 150 or 200 indels, as the number of indels within chromosome I across various *S. cerevisiae* strains was found to vary between 100 and 200. Sequence reads were simulated using *pIRS* and mapped against the reference as described previously. The *FAT-CIGAR* toolkit was utilised to anchor reads using the FAT-CIGAR string at 10, 20 and 30 bases. Variants were called from the filtered and unfiltered BAM files, processed with *vcflib* before carrying out variant classification with *vcfeval*. The classifications were corrected in order to look at the differences in the number of false positives as the stringency of read filtering increased.

4.2.4 Identifying an Optimal Read Filter

The optimal *FAT-CIGAR* read filter for improving variant calling was identified by simulating genomes to mimic the numbers of variants in chromosome I of the Bergstrom strains (see Chapter 2). As the Bergstrom strains come from five different *Saccharomyces cerevisiae* sub-populations, a representative strain was chosen from each population. The numbers of SNPs and indels in chromosome I of each strain were obtained from an average of the number of variants called by each of the graph genome software, as shown in Table 4.1. For each of the five strains, 10 genomes were simulated with the same number of SNPs and indels. The same read filtering pipeline outlined previously was utilised for sequence read simulation, mapping, read filtering, variant calling and classification. The average number of true positives, false positives and false negatives variants across the 10 genomes were calculated to compare the differences in the accuracy of the variant calls produced by each filter.

The ratio of SNPs to indels were found to be relatively constant across the Bergstrom strains despite the vast difference in range. In order to examine how the coincidence of SNPs and indels within a genome can impact read filtering, a further simulation study was carried out. 25 genomes were simulated containing 200, 400, 800, 1,600 and 3,200 SNPs and 20, 40, 80, 160 and 320 indels. The read filtering pipeline was carried out and the VCF files were compared to identify which read filter produced the most accurate

Table 4.1: **Simulating Bergstrom strains.** The five representative Bergstrom strains from each *S. cerevisiae* sub-population that were chosen for simulation, along with the average number of SNPs and indels within chromosome I of each strain, as determined through analysis using the graph genome software.

Strain	**Sub-Population**	**SNPs**	**Indels**
YPS128	North American	2,364	187
UWOPS03-461.4	Malaysian	2,265	175
DBVPG6044	West African	2,009	172
Y12	Sake	1,938	121
DBVPG6765	Wine/European	1,563	132

variant calls.

4.2.5 Read Filtering on *S. cerevisiae* Strains

Whilst studying the effect of read filtering on the accuracy of variant calling on simulated datasets was highly useful, it was also critical to study the effect of filtering on real datasets. The following three *S. cerevisiae* strains within the NCYC collection had been sequenced twice during the NCYC sequencing project: NCYC91 (haploid), NCYC87 (triploid) and NCYC1026 (tetraploid). As these strains had been re-sequenced on different sequencing plates at different times and using different sequencing library construction methods, the sequence reads could be readily compared against each other. The strain reads were mapped against the S288c reference genome with *BWA* and the resulting BAM files were filtered on both the CIGAR and FAT-CIGAR string. Variants were called from each of the filtered and unfiltered BAM files with *FreeBayes* prior to running the *vcflib* post-processing pipeline. The *bcftools isec* function was utilised to identify the number of shared and unique variants from both plates for each strain, to test whether the proportion of shared variants increased with filtering.

4.3 Results

4.3.1 Read Retention After Filtering on the FAT-CIGAR String

An initial understanding of how sequence reads are impacted by this novel method of read filtering was obtained by monitoring read retention within the BAM files. The percentage of sequence reads retained as the read filtering stringency increased is shown in Figure 4.3. As expected, the percentage of sequence reads retained was found to decrease as the number of bases on which the FAT-CIGAR string was anchored increased. Filtering reads on the minimum anchor length of five bases resulted in 94.1% read retention whilst the maximum anchor length of 30 bases resulted in 78.7% retention of reads. As the anchor length increased, the difference in percentage of reads filtered also decreased. For example, there was a 5.9% decrease in the proportion of reads from the unfiltered dataset to the BAM file filtered at five bases whilst there was a decrease of only 2.2% when moving from 25 bases to 30 bases.

4.3.2 Effect of Read Filtering on SNPs

Three genomes were simulated containing 1,000, 2,000 and 3,000 SNPs to assess whether filtering reads on the FAT-CIGAR string can improve the accuracy of the variant calls.

Figure 4.4 shows the differences in variant classification for each read filter when sequence reads were simulated in *pIRS* using the custom versus the standard error profile from the genome containing 1,000 SNPs. For both error profiles, as the stringency of filtering increased, the number of false positive variant calls also greatly decreased. Consequently, there was a slight decrease in the number of true positive variants which resulted in the number of false negative variants increasing as an artefact of removing the false positive variants.

The sequence reads simulated using the custom error profile led to a threefold greater calling of false positive variants (3,364) than true positive variants (998). Only when the reads were filtered on the FAT-CIGAR string anchored by 20 bases did the number of false positive variants (531) reduce by 84% to less than that of the true positive variants (991). There were only two false negative variant calls in the unfiltered reads which rose to nine variants when anchored stringently by 20 bases. For reads simulated using the

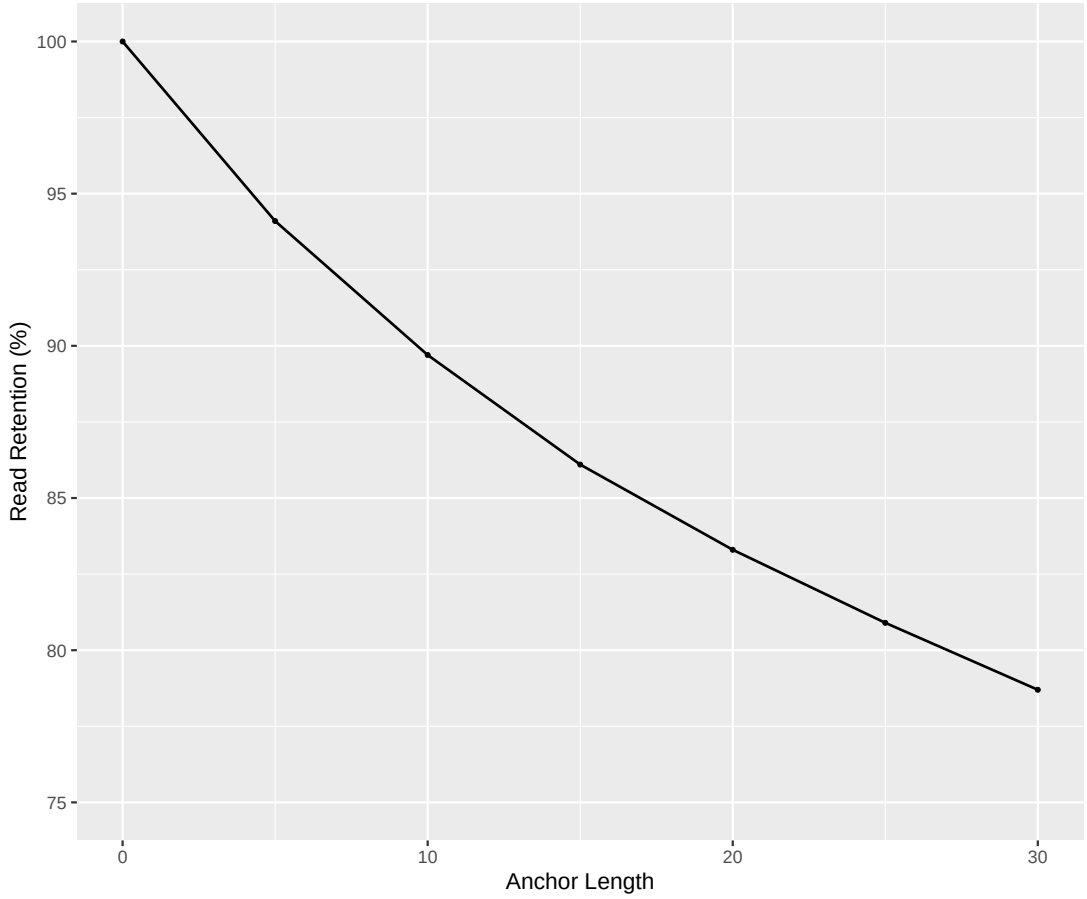


Figure 4.3: **Read Retention After Filtering.** The percentage of sequence reads that were retained in the BAM file after filtering on the FAT-CIGAR string to anchor the read ends by varying base lengths is shown in the line graph. As reads are anchored by a greater number of bases at each end, the percentage of reads that pass the filter decreases.

standard error profile, 995 variants called from the unfiltered reads were true positives whilst 388 variants were false positives. This was also found to reduce by 84% to 60 false positive variants when filtered on the FAT-CIGAR string by 20 bases and 988 true positive variants. The number of false positive variants was found to be nearly nine times greater when using the custom error profile compared to reads from the standard error profile. This was also found to be the case when comparing the error profiles from the simulated genomes containing 2,000 and 3,000 SNPs (see also Appendix Table C.1).

The increased incidence of false positive variants within the reads generated using the custom error profile also contributed to lower read retention during the filtering process. When the reads were anchored by 10 and 20 bases, this resulted in 82.3% and

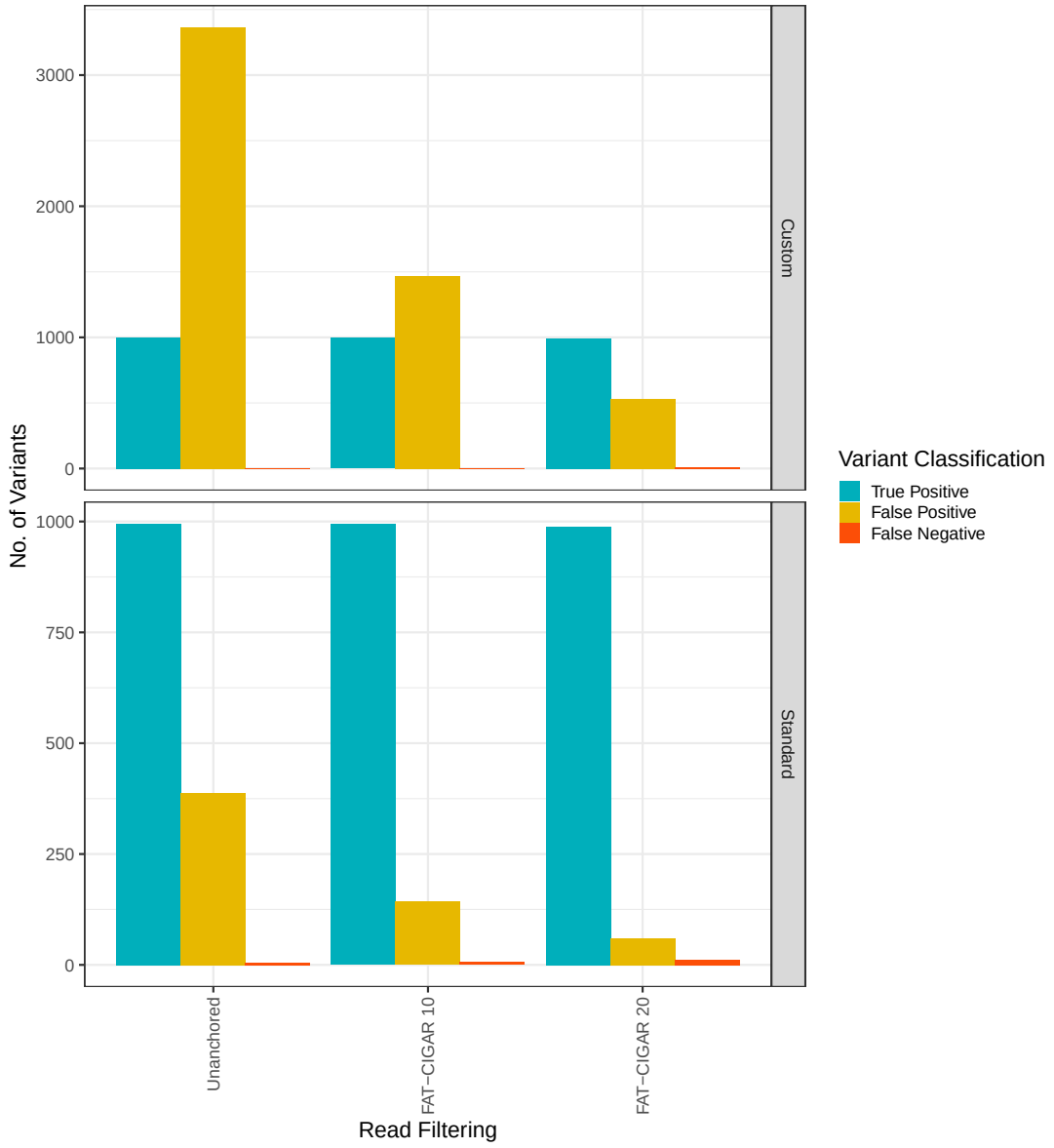


Figure 4.4: **Custom vs Standard Error Profiles.** The bar chart shows the number of true positive (blue), false positive (yellow) and false negative (orange) variant calls for the genome simulated with 1,000 SNPs, both without filtering (unanchored) and when filtered on the FAT-CIGAR string. The 10 and 20 tags refer to the base length by which the reads were anchored against the reference genome. The difference in variant calling when simulating sequence reads with the custom versus standard error profile is shown by the two boxes.

72.1% read retention when using the standard error profile and 76.8% and 58.8% for the custom error profile. Subsequently, the greater loss of sequence reads also led to an increased number of true positive variants being removed during filtering.

The default *pIRS* error profile was chosen as the most appropriate for simulating sequence reads going forward. For all three genomes, the reads were filtered on the FAT-CIGAR string and the numbers of each variant type, after correcting any misclassified variants, was obtained to understand how filtering impacts SNP calling accuracy (Figure 4.5). As with the genome containing 1,000 SNPs, as the stringency of the filter increased, the number of false positive variants was also found to greatly decrease for all genomes.

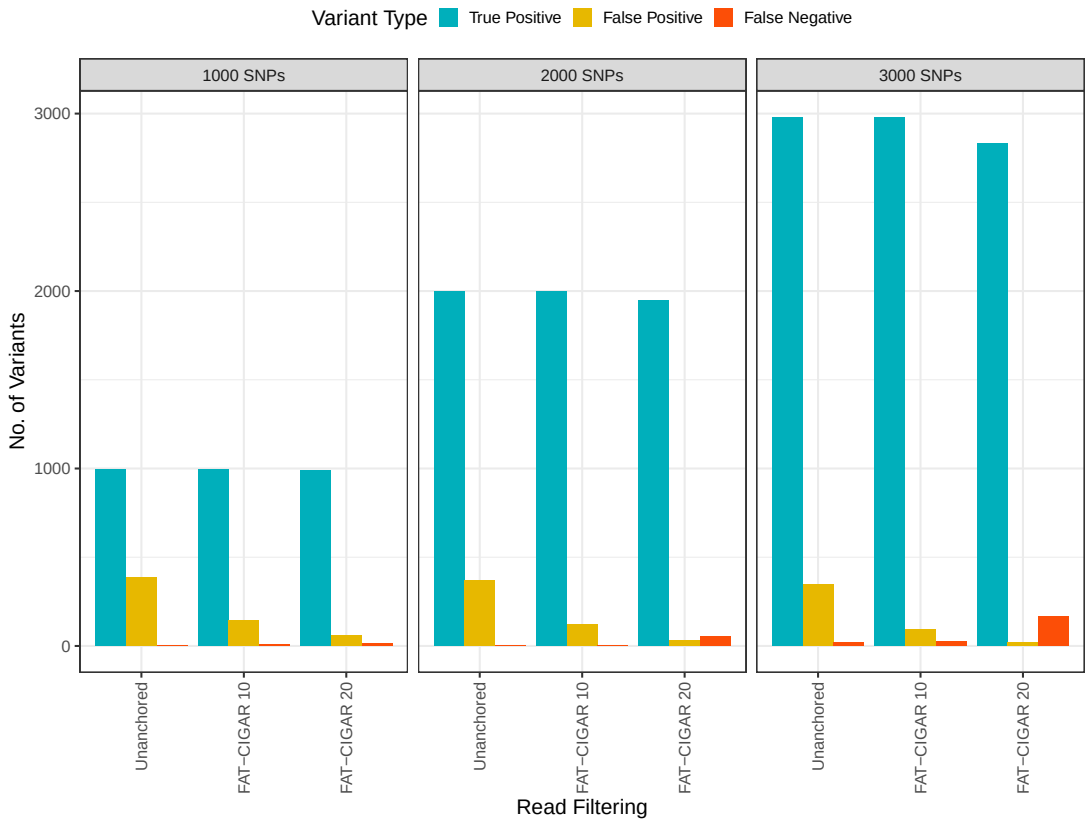


Figure 4.5: **FAT-CIGAR Filtering on SNPs.** This bar chart shows the error-corrected, classified variant calls from three simulated genomes containing 1,000, 2,000 and 3,000 SNPs. The sequence reads were simulated using the default error profile in *pIRS*. For each genome, the number of variants that were classified as true positives (blue), false positives (yellow) and false negatives (orange) has been shown for both unfiltered reads and reads filtered on the FAT-CIGAR string.

There were also fewer false positive variants in the unfiltered calls from the 3,000 SNPs (349) genome compared to the 1,000 SNPs genome (388). The greatest reduction in false positive variant calls was found to be between the unfiltered sequence reads and reads filtered on the FAT-CIGAR string anchored by 10 bases by 63% to 142, 67% from 368 to 122, 74% to 91 for the 1,000, 2,000 and 3,000 SNPs genomes, respectively. The

number of true positive variant calls also remained relatively similar as there were only one to three fewer true positives across the three genomes. Increasing the anchor length to 20 bases resulted in a further decrease in false positive variants by 21% to 60 and 17 for the 1,000 and 3,000 SNPs genome, respectively and by 24% to 32 for the 3,000 SNPs genome. However, the 20 bp anchor length was found to be too stringent as it also resulted in the filtering of a greater number of true positive variants, particularly for the 3,000 SNPs genome. The number of true positive variants was found to decrease by 7 (0.8%) for the 1,000 SNPs genome, 50 (2.5%) for the 2,000 SNPs genome and 146 (5%) for the 3,000 SNPs genome. Consequently, there was a proportionate increase in the number of false negative variant calls across all genomes.

4.3.3 Effect of Read Filtering on Indels

Read filtering was carried out on both CIGAR and FAT-CIGAR strings for four genomes simulated with 1,000 SNPs and varying number of indels, as shown in Figure 4.6. As the number of indels within the genome increased, the number of true positive SNP calls was found to decrease from 752 for the 100 indels genome to 731 for the 300 indels genome and the number of false positive SNP calls was found to increase from 15 to 34, respectively, despite containing the same number of SNPs. Conversely, the number of false negative SNP calls was found to increase as the number of variants within the genome increased.

Both the CIGAR and FAT-CIGAR filtering was found to improve the accuracy of variant calling by reducing the percentage of false positive variant calls. Filtering on the CIGAR string had a slightly greater impact on the accuracy of indels than SNPs. Anchoring the reads at 30 bases decreased the percentage of false positive SNP calls from the unfiltered datasets by 15-40% across the four genomes and the percentage of indels by 11-54%. This also resulted in a slight reduction in the percentage of true positive variant calls by 0-0.1% for SNPs and 0-1.5% for indels in spite of the stringent anchor length.

Reads filtered on the FAT-CIGAR string produced more accurate variant calls than reads filtered on the CIGAR string. Filtering reads on the FAT-CIGAR string by 10 bases was shown to remove a greater number of false positive variants than filtering

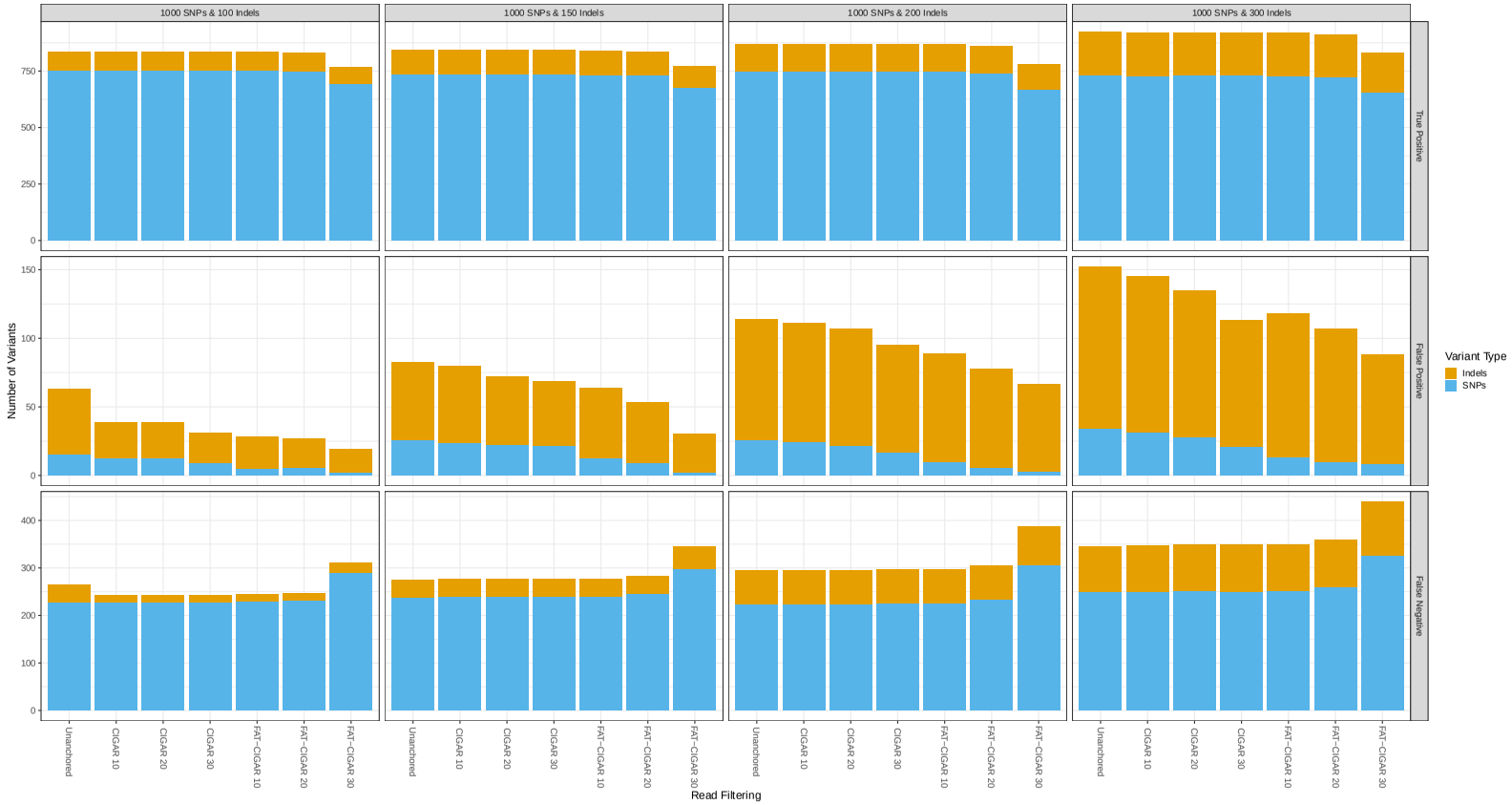


Figure 4.6: **Read Filtering on Indels.** This bar chart shows the classified variant calls from four genomes simulated with 1,000 SNPs and 100, 150, 200 and 300 indels, respectively. Read filtering was carried out using both the CIGAR and FAT-CIGAR strings and filtered at 10, 20 and 30 bases. The blue bars denote the numbers of SNPs and the orange bars denote the numbers of indels.

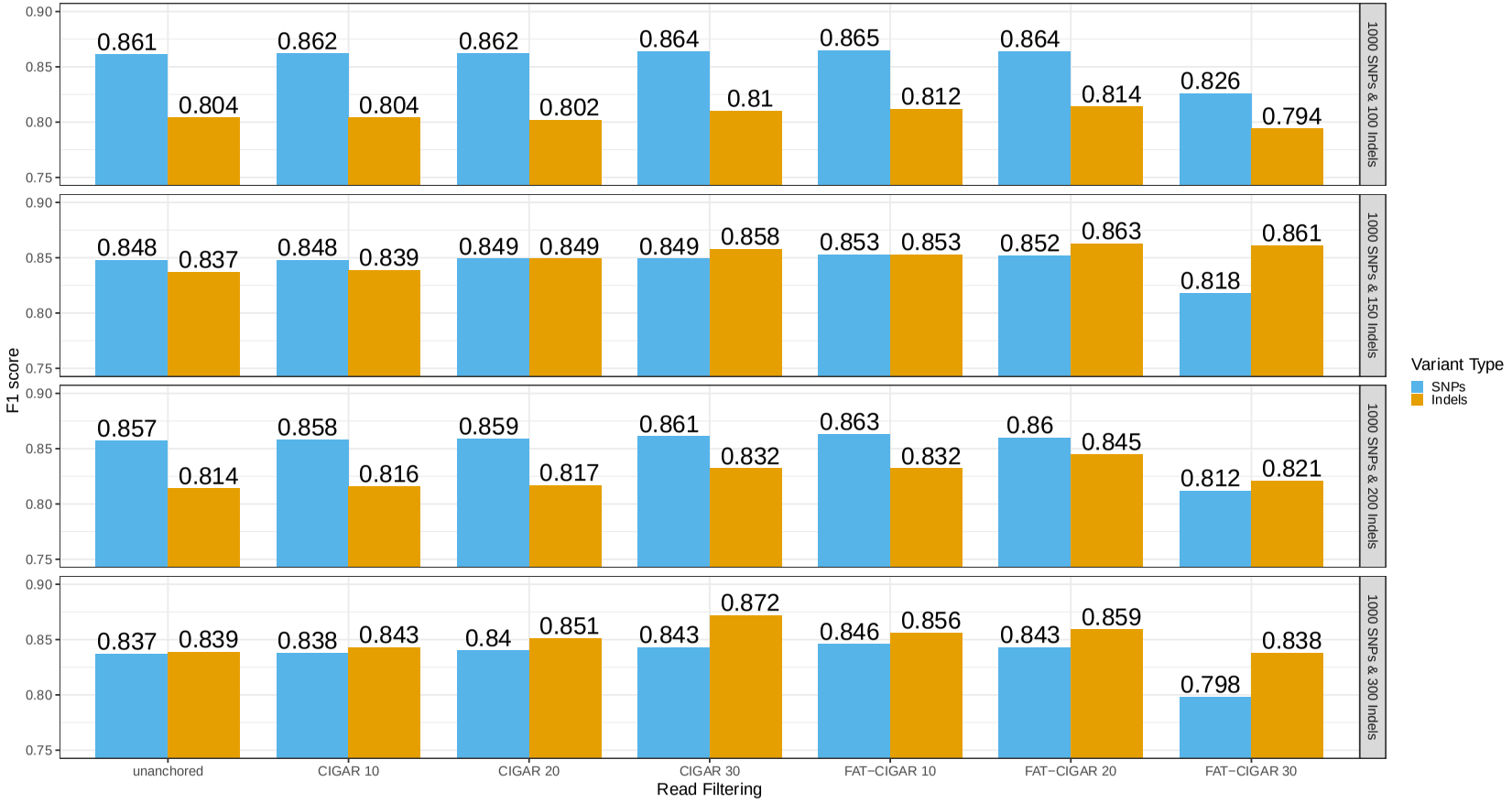


Figure 4.7: **F1 Score Comparing Filtering Accuracy.** F1 scores comparing the accuracy of the SNP (blue) and indel (orange) variant calls after filtering on the CIGAR and FAT-CIGAR strings for all four genomes has been shown in the bar chart.

on the CIGAR string by 30 bases due to reducing false positive SNP calls by 50-67% from the unfiltered reads. The only exception was the 300 indels genome in which filtering by 30 bases on the CIGAR string produced fewer false positive variants as a greater number of false positive indels were removed (26) compared to reads filtered on the FAT-CIGAR string by 10 bases (13). FAT-CIGAR anchoring of reads by 30 bases greatly reduced the percentage of false positive SNP calls from the unfiltered reads by 76-92% and indel calls by 27-65%. However, the stringency of this filter also resulted in a greater reduction in the percentage of true positive variant calls by 8-11% for SNPs and by 8.5-9.4% for indels.

The accuracy of the read filters was determined by evaluating the trade-off between the true positives, false positives and false negative variant calls using the F1 score. Figure 4.7 demonstrates the differences in the F1 score for SNPs and indels after filtering for all four genomes. There was a clear distinction shown between the optimal read filter for improving variant calling in SNPs and indels. Due to the consistency of the SNPs across all four genomes, filtering on the FAT-CIGAR string by 10 bases produced the most accurate calls. Filtering on the CIGAR string up to 30 bases also incrementally increased the F1 scores of the SNP calls but extending the anchor length beyond 10 bases on the FAT-CIGAR string reduced the accuracy of the calls. As the number of indels increased, the most optimal filter for calling indels was found to shift towards the CIGAR string. For both the 100, 150 and 200 indels genomes, filtering on the FAT-CIGAR string by 20 bases was found to produce the greatest F1 scores. However, for the genome containing 300 indels, filtering on the CIGAR string by 30 bases produced a much greater F1 score, 0.872 compared to filtering on the FAT-CIGAR string by 20 bases, 0.859. Anchoring reads by 30 bases on the FAT-CIGAR was found to produce the calls with the lowest accuracy, albeit removing the highest number of false positive variants, with the F1 scores dropping to that of below the unfiltered reads. The combined variant calls revealed that reads filtered on the FAT-CIGAR string by 10 bases had the highest F1 score across all the genomes.

4.3.4 Read Filtering on Realistic Datasets

In order to evaluate how read filtering can impact variant calling when mimicking the number of variants found in real datasets, genomes were simulated with the same num-

ber of SNPs and indels as within chromosome I of five Bergstrom strains from varying *S. cerevisiae* sub-populations. For each strain, 10 genomes were simulated and the differences in the average number of classified variants before and after filtering has been shown in Figure 4.8 (see also Appendix Table C.2). The average percentage of true positive variant calls was found to be slightly low across the five strains, with only 51-75% of the SNPs within the genome called and 63-84% of the indels called. In addition, the average number of false positive variant calls was found to be lower for SNPs (17-27) than for indels (31-45) for all strains despite the much lower ratio of indels within the genome.

As seen previously, filtering on both the CIGAR and FAT-CIGAR strings reduced the percentage of false positive SNP and indel calls with filtering on the FAT-CIGAR string removing a greater number of false positive SNPs. Filtering on the CIGAR string by 30 bases reduced the average percentage of false positive SNP calls by 28-35% and indel calls by 33-45% from the unfiltered reads. The percentage of true positive SNP calls also decreased slightly by 0-0.17% and indel calls by 0.9-1.96%. Filtering on the CIGAR string by 30 bases was also able to reduce more false positives on average than filtering on the FAT-CIGAR string by 10 bases due to the stringency of the 30 bp anchor length removing 7, 9 and 7 fewer false positive indel calls for UWOPS03-461.4, DBVPG6044 and DBVPG6765, respectively. For YPS128 and Y12, filtering on the FAT-CIGAR string by 10 bases produced a slightly lower average number of false positive calls. However, of the 10 individual genomes for both strains, five and six of the genomes for Y12 and YPS128, respectively, had lower false positive calls when filtered on the CIGAR string by 30 bases. Reads filtered on the FAT-CIGAR string anchored by 30 bases had the lowest number of false positive variants, with 76-85% fewer false positive SNP calls and 55-60% fewer indel calls than the unfiltered reads. However, the stringency of the filter also resulted in a sharp decrease in the average percentage of true positive variants calls by 14-21% for SNPs and 15-22% for indels.

The differences in the accuracies of the unfiltered and filtered variant calls for all five strains were compared using the F1 score as depicted in Figure 4.9. The F1 scores were also found to be relatively similar across all five strains. For SNP calling, as with the previously simulated genomes, filtering reads on the FAT-CIGAR string by 10 bases was found to produce the most accurate variant calls. The strains DBVPG6044 and

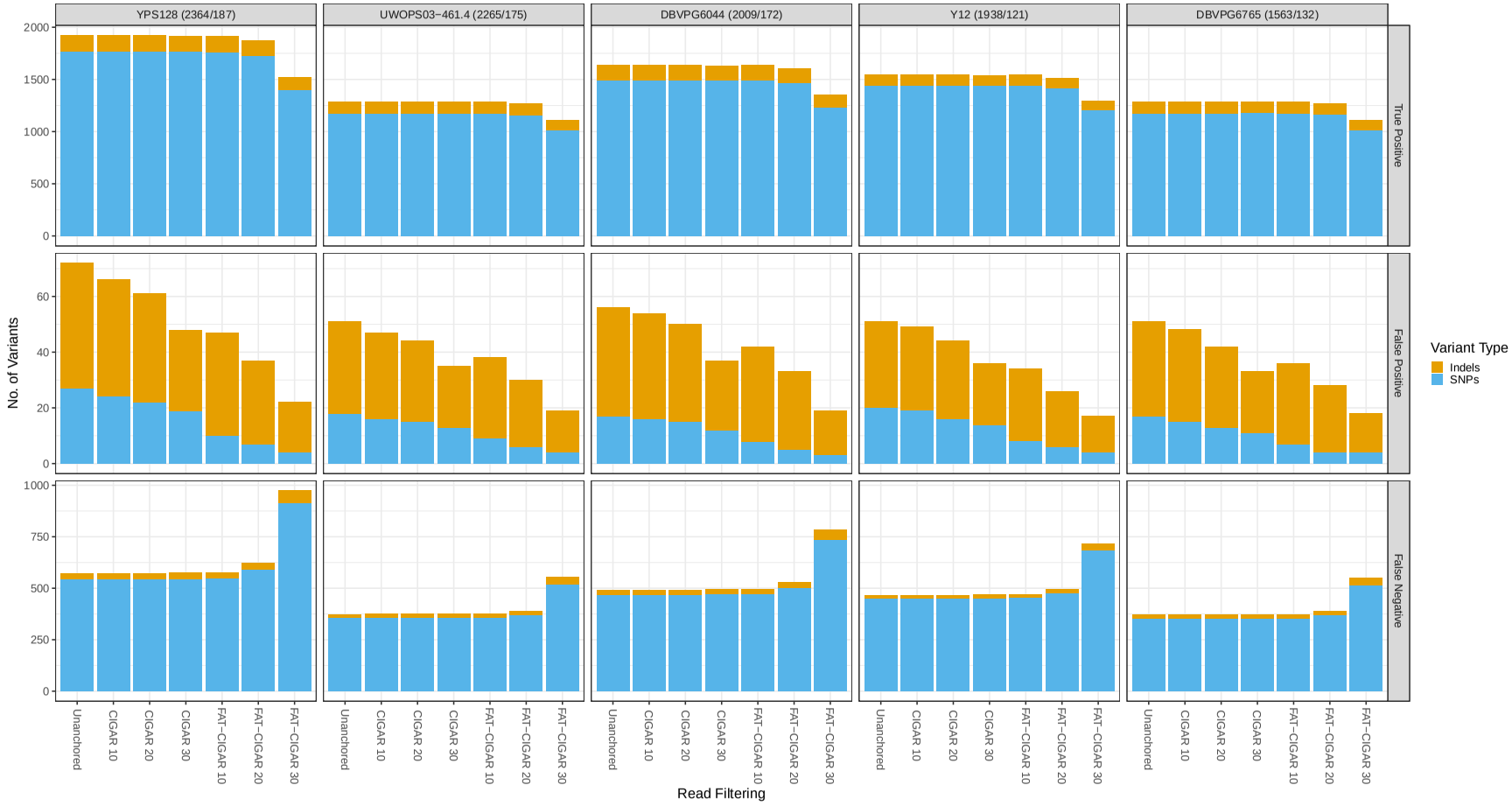


Figure 4.8: **Read Filtering on Simulated Bergstrom Strains.** This bar chart shows the average numbers of variants called from across 10 simulated genomes for each of the five strains. The numbers of SNPs and indels simulated are shown adjacent to the strain name at the top of each column.

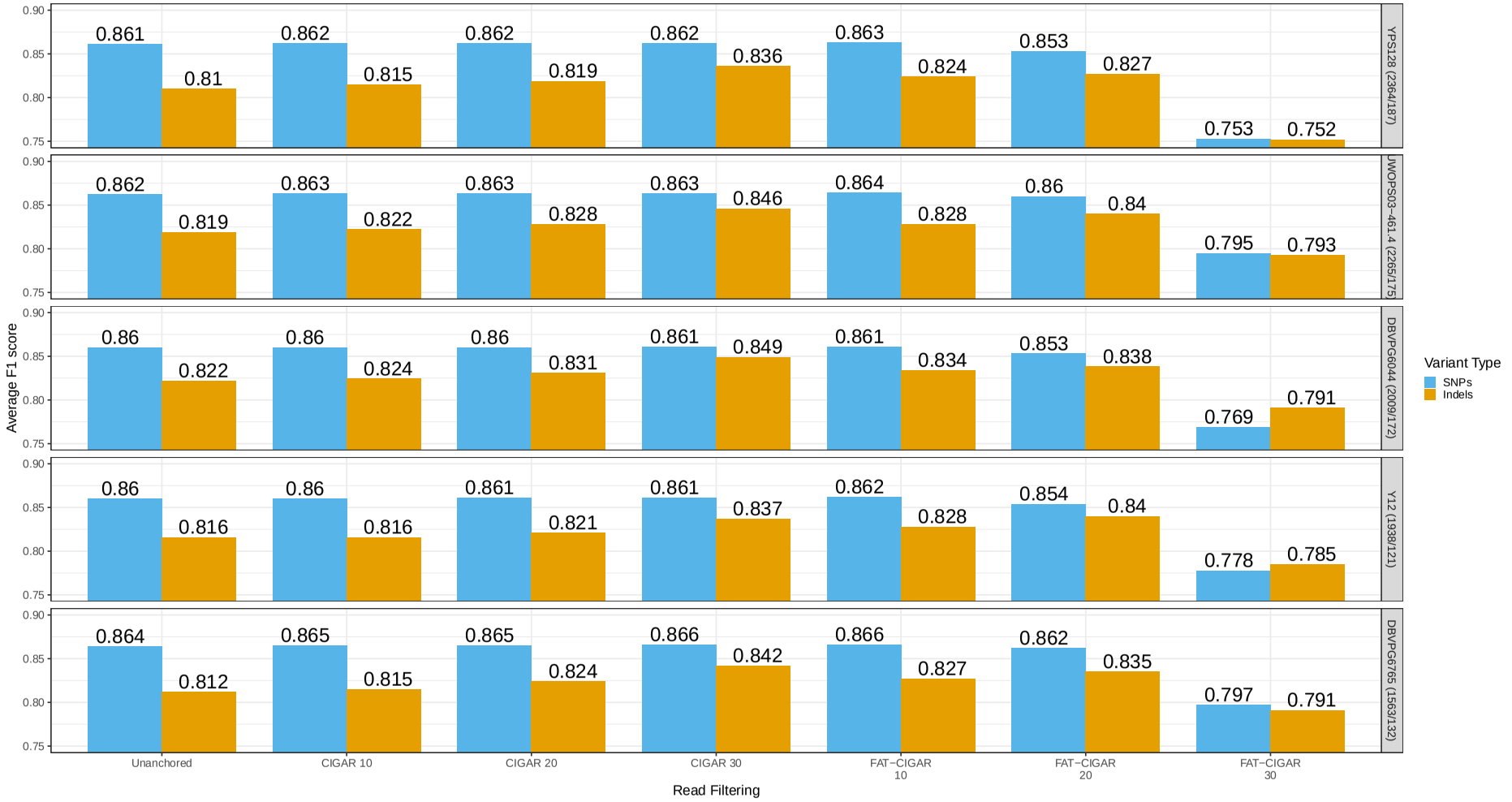


Figure 4.9: **Bergstrom Strains F1 Scores.** The average F1 scores comparing the accuracy of variant calling SNPs and indels after filtering for the five Bergstrom strains has been shown in the bar chart.

DBVPG6765 had similar F1 scores, 0.861 and 0.866, for reads filtered on the CIGAR string by 30 bases. However, CIGAR 30 filtering produced slightly higher numbers of false positives in comparison to the reads filtered on the FAT-CIGAR string. In concordance with the results seen in Figure 4.8, increasing the anchor length beyond 10 bases caused a rapid decline in variant calling accuracy. The stringency of FAT-CIGAR filtering at 30 bases caused the F1 score to be much lower than that of the unfiltered reads.

The most accurate indel calls for four of the strains were produced from reads filtered on the CIGAR string by 30 bases followed by reads filtered on the FAT-CIGAR string by 20 bases. The only exception to this observation was Y12, in which reads filtered on the FAT-CIGAR by 20 bases produced the highest F1 score as on average it called two fewer false positives and the same number of true positive variants. The F1 scores, looking at the overall accuracy of both SNPs and indels together, showed that reads filtered by 30 bases on the CIGAR string produced the most accurate variant calls for all five strains. For Y12 and YPS128, reads filtered on the FAT-CIGAR string by 10 bases also had the same accuracy, 0.86. The accuracy of variant calling when combining SNPs filtered on the FAT-CIGAR string by 10 bases and indels filtered on the CIGAR string by 30 bases was also examined (see also Appendix Figure C.1). The F1 score was found to remain the same as that of reads filtered on the CIGAR string by 30 bases for DBVPG6044, DBVPG6765 and Y12. There was an 0.001 increase in the F1 scores for UWOPS03-461.4 and YPS128.

A further 25 genomes were simulated to examine the impact of read filtering on variant calling accuracy in the presence of varying ratios of SNPs and indels. Across all of the genomes, as the number of variants within the genome increased, the percentage of false positive variant calls was found to decrease (Figures 4.10-4.14). However, as the ratio of SNPs to indels decreased across genomes containing the same number of SNPs, the number of false positives was found to increase. As expected, the overall numbers of true positive calls were found to increase as the ratio of SNPs to indels decreased in genomes containing the same numbers of SNPs due to the higher numbers of indels within the genome. The number of true positive SNP calls was found to remain relatively similar across the unfiltered reads from genomes with the same numbers of SNPs. However, as the ratio of SNPs to indels decrease, the overall number of false

negative variant calls from the unfiltered reads was found to increase due to the increase in false negative indel calls. As the stringency of filtering increased, the number of false positives SNP and indel calls were shown to decrease. There was a sharper decrease seen in the false positive SNP calls when filtered on the FAT-CIGAR string than the CIGAR string. The first five genomes containing 200 SNPs (see Figure 4.10) showed that filtering reads on the FAT-CIGAR string by 10 bases also produced a lower number of false positive calls in comparison to reads filtered on the CIGAR string by 30 bases for the genomes containing 20, 40 and 80 indels. The genome containing 160 and 320 indels produced fewer overall false positives when filtered on the CIGAR string by 30 bases, likely due to the increased incidence of indels within the genome. This was also found to be the case for the 400 SNPs genomes (Figure 4.11), the 800 SNPs (Figure 4.12) genome containing 80, 160 and 320 indels, 1,600 SNPs (Figure 4.13) genome containing 320 indels and 3,200 SNPs genome (Figure 4.14) containing 80, 160 and 320 indels.

F1 scores were obtained for the SNP and indel calls in each of the 25 genomes to identify which read filter produced the highest score. The most optimal filter for producing accurate SNP calls from each genome is shown in the heat map in Figure 4.15a. For genomes containing a lower incidence of SNPs and indels (i.e. top left corner), filtering on the FAT-CIGAR string by 10 bases produced the most accurate SNP calls. As the ratio of SNPs to indels decreased, increasing the anchor length to a more conservative filter of 20 bases was preferred (i.e. top right corner). However, as the number of SNPs within the genome increased, less stringency in filtering was favoured (i.e. bottom left corner). Reducing the anchor length and filtering on the CIGAR string tended to produce more accurate calls. The increased number of SNPs also resulted in the same F1 scores across the different filters, particularly for the CIGAR string. For example, for the 3,200 SNPs genome, anchoring reads by 10, 20 and 30 bases on the CIGAR string produced the same F1 scores for all but the 320 indels genome. With indel calling (see Figure 4.15b), reads that were filtered highly conservatively on the FAT-CIGAR string by 30 bases produced the highest F1 scores for genomes with the fewest number of variants. Both as the number of SNPs and the number of indels increased across the genomes, less stringency in filtering was preferred with F1 scores shifting towards favouring the CIGAR string. For the majority of genomes, filtering on the CIGAR string by 30 bases produced the most accurate indels calls.

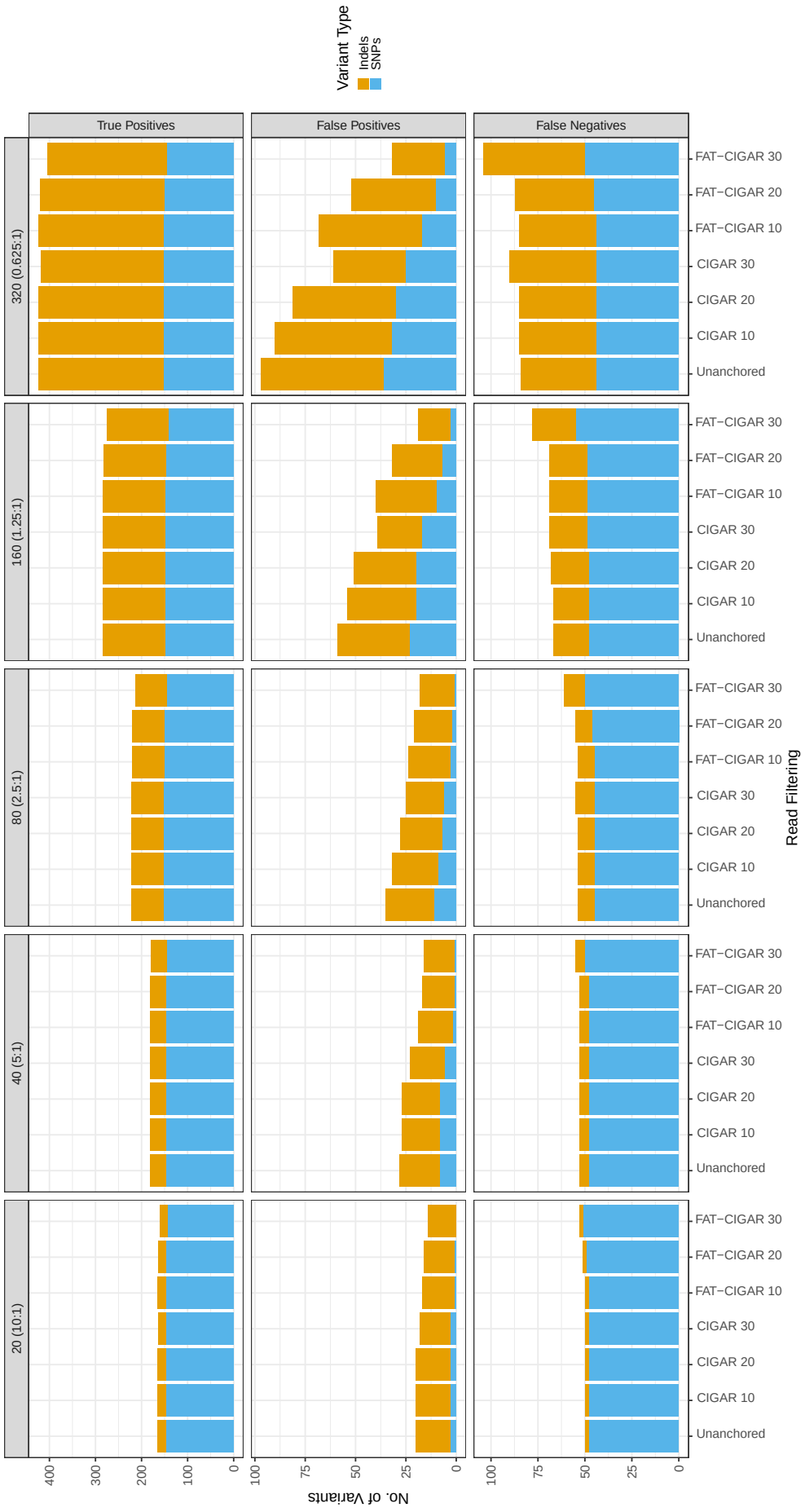


Figure 4.10: **200 SNPs Genome.** This bar chart shows the percentage of variants called after filtering from five genomes containing 200 SNPs and 20, 40, 80, 160 and 320 indels.

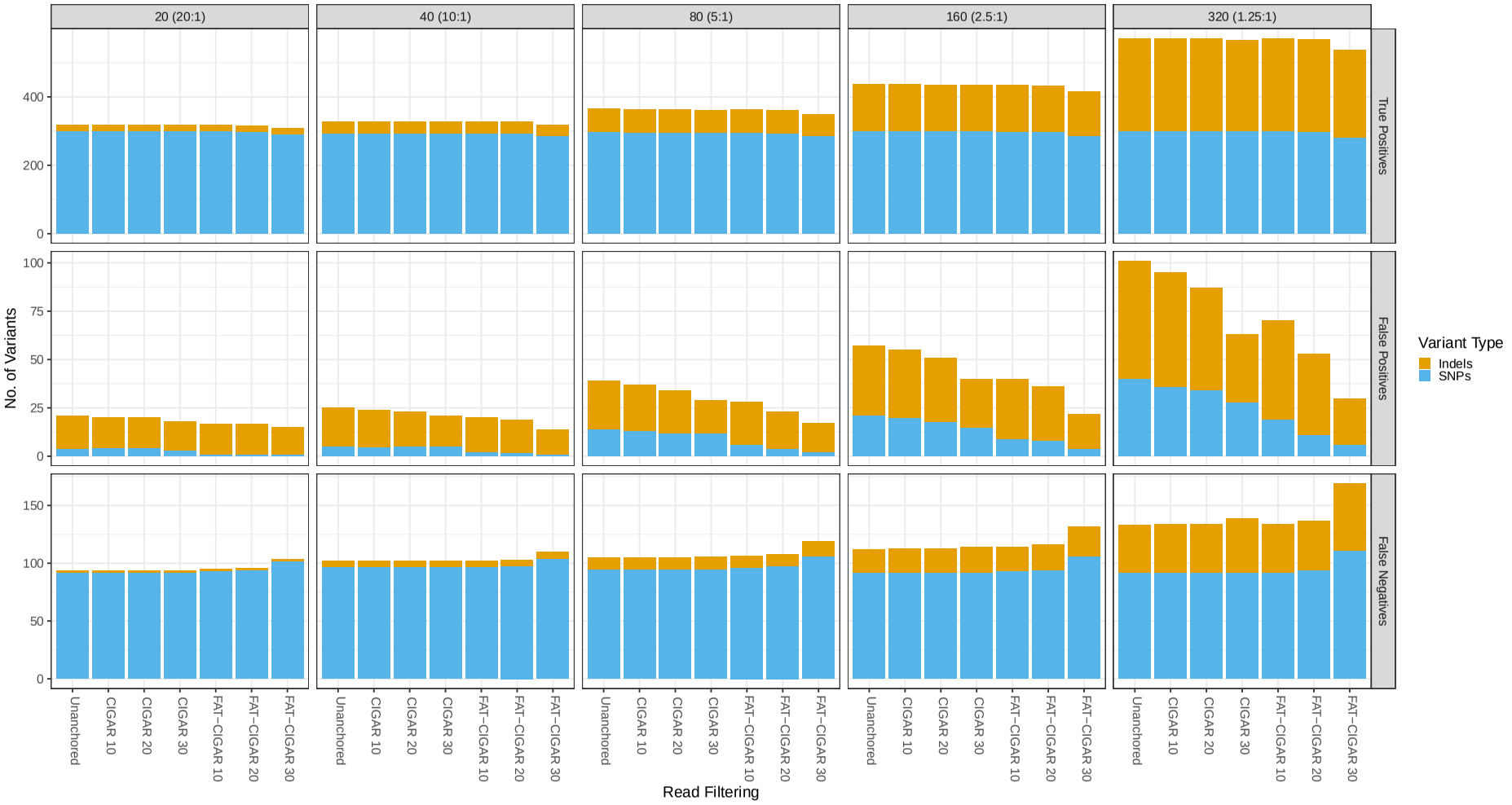


Figure 4.11: **400 SNPs Genome.** This bar chart shows the percentage of variants called after filtering from five genomes containing 400 SNPs and 20, 40, 80, 160 and 320 indels.

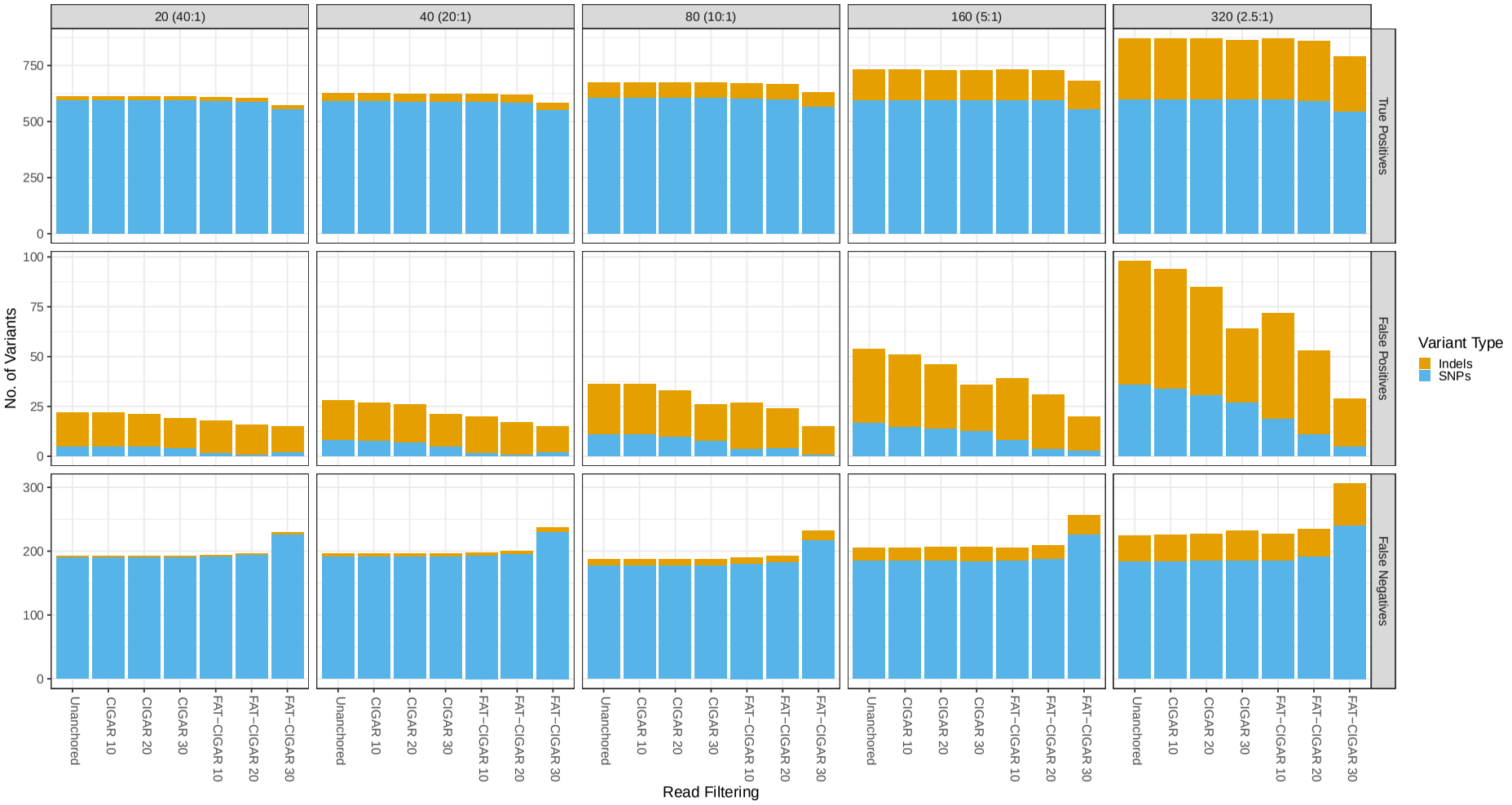


Figure 4.12: **800 SNPs Genome.** This bar chart shows the percentage of variants called after filtering from five genomes containing 800 SNPs and 20, 40, 80, 160 and 320 indels.

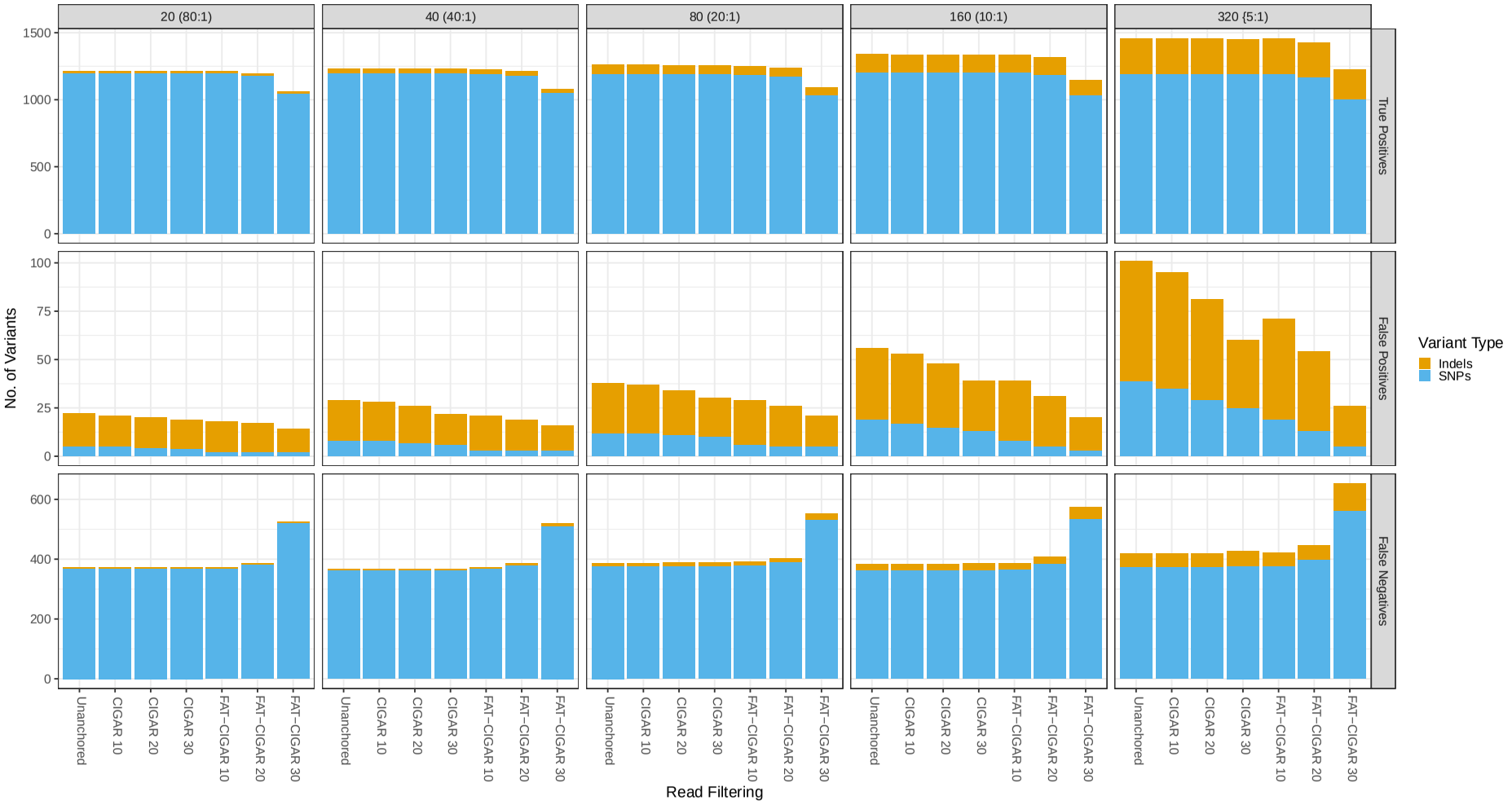


Figure 4.13: **1600 SNPs Genome.** This bar chart shows the percentage of variants called after filtering from five genomes containing 1,600 SNPs and 20, 40, 80, 160 and 320 indels.

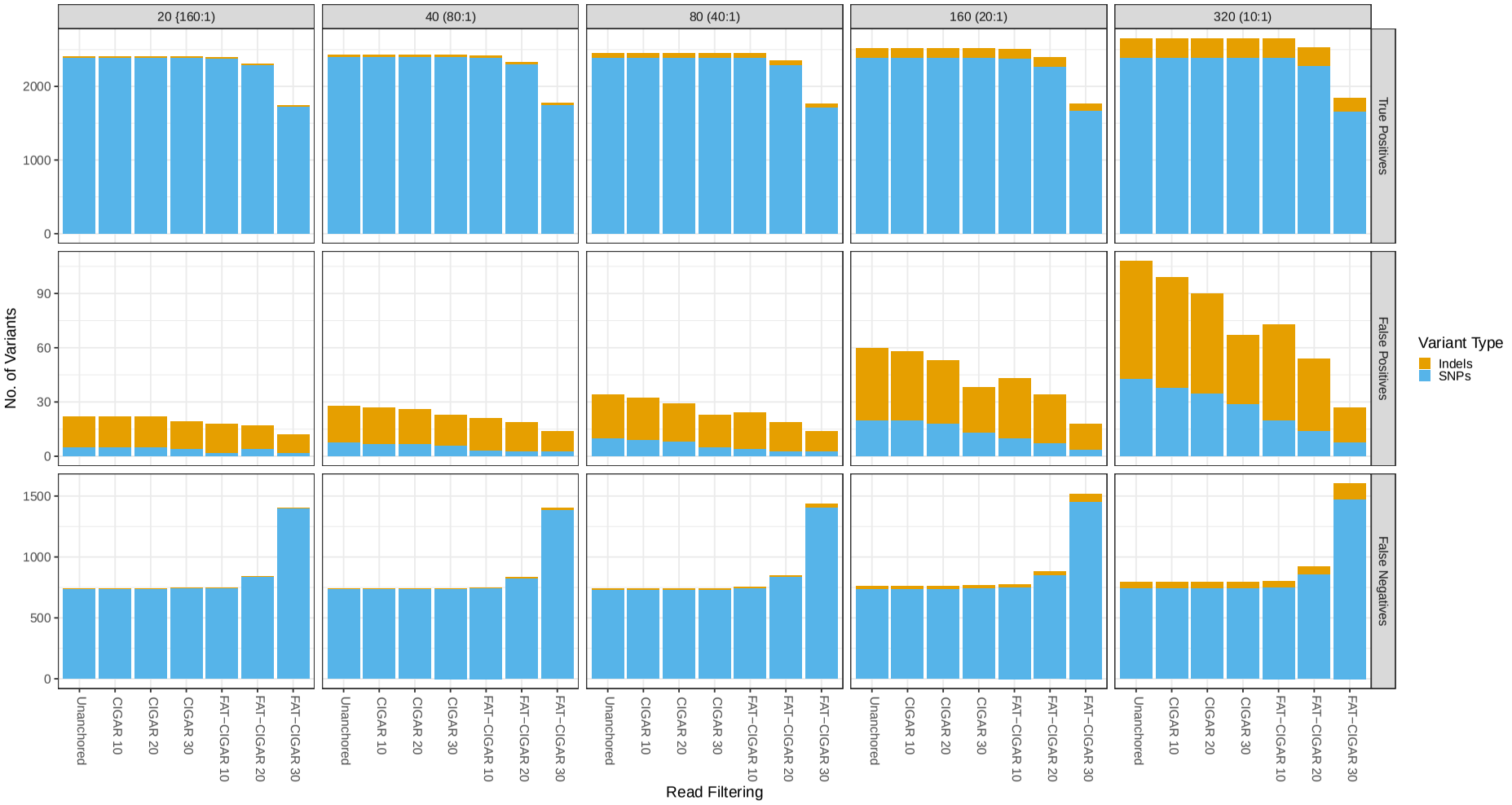


Figure 4.14: **3200 SNPs Genome.** This bar chart shows the percentage of variants called after filtering from five genomes containing 3,200 SNPs and 20, 40, 80, 160 and 320 indels.

		Indels				
		20	40	80	160	320
SNPs	200	F-10	F-20	F-10/F-20	F-20	F-20
	400	F-10	F-10	F-10	F-10/F-20	F-20
	800	C-30	F-10	F-10	F-20	F-10/F-20
	1600	C-20/C-30	C-20/C-30/F-10	C-20/C-30/F-10	C-30/F-10	F-10
	3200	C-10/C-20/C-30	C-10/C-20/C-30	C-10/C-20/C-30	C-10/C-20/C-30	F-10

(a) Optimal Read Filter for SNPs

		Indels				
		20	40	80	160	320
SNPs	200	F-30	F-30	F-20	F-30	C-30/F-30
	400	F-30	F-30	C-30	F-30	C-30
	800	C-30/F-20/F-30	C-30/F-20	C-30	C-30	C-30/F-20
	1600	C-30/F-20/F-30	C-30	C-30	C-30	C-30
	3200	F-20	C-30	C-30	C-30	C-30

(b) Optimal Read Filter for Indels

Figure 4.15: **Heat Map of Optimal Read Filters.** This heat map shows the read filter that produced the most accurate SNP and indel calls, as determined based on the F1 scores, for all 25 simulated genomes. C stands for CIGAR string and F for FAT-CIGAR string, with the adjacent number indicating the length of base anchoring used by the filter.

4.3.5 Filtering NCYC Strains

The effect of read filtering on real strains was assessed by comparing the proportion of shared variants between the same strains that were sequenced twice on different 96 well sequencing plates and under different conditions (e.g. different sequencing library techniques). Three *S. cerevisiae* strains from the NCYC collection that had varying levels of ploidy were chosen for this comparison. The percentage of unique and shared variants from reads sequenced on Plate 1 and Plate 10 for the haploid strain NCYC91 is shown in Table 4.2. Both plates were found to have a high percentage of shared SNPs,

90.85% and shared indels, 86.46% using the unfiltered reads. Filtering the reads was found to increase the percentage of shared variants between both plates. The greatest proportion of shared SNPs was found from reads filtered on the FAT-CIGAR string by 20 bases. This increased the percentage of SNPs by 3.92% for the unfiltered calls, 0.91% for calls filtered on the allelic fraction (AF) to be greater than 0.2 and 3.32% for calls filtered on AF $>$ 0.9. The allelic fraction is the frequency of the allele observed divided by the read depth at that locus. The optimal AF threshold for filtering SNPs was found to be 0.2 as it allowed for the removal of variants with low read support that have a greater likelihood of being false positive thus increasing the proportion of shared variants across both the anchored and unanchored reads. Increasing the AF threshold to 0.9 was found to slightly reduce the proportion of shared SNPs as the threshold is quite conservative resulting in the removal of true variants without a high read support. Extending the anchor length to 30 bases was found to be too stringent resulting in a fewer percentage of shared variants.

For indels, filtering on the CIGAR string by 30 bases on the unconstrained dataset led to the highest percentage of shared variants, 90.3%. For indel calls filtered on AF $>$ 0.2, filtering on the CIGAR string by 20 bases increased the percentage of shared indels by 0.92%. In concordance with the SNP calls, utilising an AF threshold of 0.2 produced the highest proportion of shared indels between both plates. Increasing the filtering threshold to 0.9 was found to greatly reduce the proportion of shared indels likely as a result of a higher number of true indels being removed due to fewer indels having as high a read support. However, for the dataset filtered on AF $>$ 0.9, there was a much greater increase in the percentage of shared variants when filtered on the FAT-CIGAR string than on the CIGAR string. The highest percentage of shared indel calls was found when reads were anchored by 20 bases, resulting in an increase of 10.1%.

The percentage of unique and shared variants called from reads sequenced on Plates 1 and 10 for the triploid strain NCYC87 is shown in Table 4.3. The variant calls were also filtered on the allelic fraction into equal sized bins with the bins centred on expected frequencies of heterozygous variants as determined by the ploidy level (i.e. 0.333, 0.667 and 1.000 for a triploid genome). The bins containing the heterozygous variants were then combined into a single bin for variants with AF between 0.16 and 0.84. Filtering the sequence reads by 20 bases on the FAT-CIGAR string increased the percentage of

Table 4.2: **NCYC91.** The proportion of shared SNPs and indels both before and after filtering NCYC91 sequence reads derived from NCYC sequencing Plates 1 and 10. The unfiltered variant calls were compared to variants filtered on the allelic fraction at greater than 0.2 and 0.9. P1 refers to the percentage of variants unique to Plate 1, Both refers to variants shared by both plates and P10 refers to variants that are unique to Plate 10. AF refers to the allelic fraction. The highest percentages of shared variants are highlighted in red.

Read	**All Frequencies**			**AF $>$ 0.2**			**AF $>$ 0.9**		
Anchoring	**P1**	**Both**	**P10**	**P1**	**Both**	**P10**	**P1**	**Both**	**P10**
SNPs									
unfiltered	3.97	90.85	5.18	1.71	95.23	3.06	6.94	91.61	1.45
CIGAR 10	4.09	91.08	4.83	1.85	95.28	2.87	6.21	92.32	1.47
CIGAR 20	4.08	91.11	4.82	1.88	95.22	2.90	6.20	92.31	1.50
CIGAR 30	4.14	91.08	4.78	1.92	95.14	2.94	6.30	92.17	1.53
FAT-CIGAR 10	2.66	93.77	3.57	1.38	96.14	2.48	4.12	94.40	1.48
FAT-CIGAR 20	2.11	**94.77**	3.12	1.40	**96.15**	2.45	3.13	**94.93**	1.94
FAT-CIGAR 30	2.74	92.74	4.53	2.43	93.42	4.15	4.01	92.25	3.74
Indels									
unfiltered	5.57	86.46	7.97	4.11	90.82	5.07	32.74	66.01	1.25
CIGAR 10	6.02	87.16	6.82	4.82	90.59	4.58	31.39	67.36	1.24
CIGAR 20	4.78	89.43	5.79	4.17	**91.75**	4.08	32.65	66.06	1.30
CIGAR 30	4.41	**90.30**	5.28	4.08	91.44	4.48	36.79	61.66	1.55
FAT-CIGAR 10	5.69	87.45	6.86	4.53	91.05	4.43	24.15	74.32	1.53
FAT-CIGAR 20	5.61	88.12	6.27	4.89	90.42	4.70	21.31	**76.15**	2.54
FAT-CIGAR 30	6.72	86.12	7.17	6.42	87.29	6.30	22.14	73.75	4.11

shared SNPs by 2.36% for the unfiltered calls whilst filtering on the CIGAR string by 20 bases increased the percentage of indels by 2.45%. As expected, the dataset filtered on AF $<$ 0.16 had the lowest proportion of shared variants as the majority of variants retained were likely to be false positives thus only unique to one plate. Consequently, it had the greatest increase in the percentage of shared variants for SNPs from reads filtered on the FAT-CIGAR string by 30 bases, 14.19% and indels filtered on the CIGAR string by 30 bases, 15.3%. The percentage of shared heterozygous SNPs only increased

Table 4.3: **NCYC87.** The proportion of shared SNPs and indels both before and after filtering NCYC87 sequence reads derived from NCYC sequencing Plates 1 and 10. The unfiltered variant calls were compared to variants filtered on the allelic fraction. The highest percentages of shared variants are highlighted in red.

Read	All Frequencies			AF < 0.16			0.16 <= AF < 0.84			AF >= 0.84		
Anchoring	P1	Both	P10	P1	Both	P10	P1	Both	P10	P1	Both	P10
SNPs												
unfiltered	4.66	92.12	3.22	39.07	24.02	36.92	4.11	93.17	2.72	2.62	95.96	1.42
CIGAR 10	4.86	92.11	3.03	41.54	23.21	35.25	4.14	**93.19**	2.67	2.68	95.79	1.53
CIGAR 20	4.94	92.07	2.99	41.83	22.84	35.33	4.31	92.99	2.70	2.75	95.67	1.58
CIGAR 30	5.09	92.03	2.88	42.38	23.49	34.13	4.63	92.49	2.88	2.84	95.47	1.69
FAT-CIGAR 10	3.66	94.47	1.87	35.44	23.51	41.05	5.74	91.53	2.73	2.37	**96.41**	1.22
FAT-CIGAR 20	4.10	**94.48**	1.43	32.66	33.89	33.45	10.71	82.65	6.65	2.68	95.97	1.35
FAT-CIGAR 30	6.94	90.64	2.42	34.46	**38.21**	27.33	15.89	70.88	13.23	5.09	93.19	1.72
Indels												
unfiltered	8.11	86.06	5.82	39.34	16.90	43.76	10.07	**82.13**	7.80	8.56	88.56	2.87
CIGAR 10	7.87	87.22	4.91	42.95	14.85	42.20	10.34	81.56	8.09	8.72	88.03	3.25
CIGAR 20	7.50	**88.51**	3.99	37.60	18.24	44.16	13.20	77.81	8.99	9.11	87.27	3.62
CIGAR 30	8.66	87.32	4.03	34.58	**32.21**	33.21	17.63	69.07	13.30	10.89	84.12	5.00
FAT-CIGAR 10	9.37	85.95	4.68	37.59	20.39	42.01	13.42	79.30	7.28	7.46	**88.62**	3.92
FAT-CIGAR 20	11.18	84.48	4.35	40.25	24.93	34.82	17.46	72.14	10.40	9.51	85.67	4.82
FAT-CIGAR 30	15.25	80.03	4.72	41.56	29.56	28.88	22.41	64.05	13.54	14.15	80.10	5.75

by 0.02% when filtered on the CIGAR string by 10 bases. However, for the indel calls, read filtering was found to be too conservative causing the proportion of shared variants to decrease. The proportion of shared SNPs was also found to increase from the unfiltered dataset for the heterozygous bins whilst the proportion of shared indels was found to decrease. The homozygous variant calls (i.e. AF $>$ 0.84) had the highest proportion of shared variants across all of the datasets, except for indels filtered on the CIGAR string by 30 bases. This is likely due to the presence of fewer homozygous variants as NCYC87 is a triploid strain thus the majority of variants fall within the heterozygous bins and the high threshold for read support ensuring the filtering of false positive variant calls. Read filtering within this dataset only caused a slight increase in the proportion of shared variants with reads anchored on the FAT-CIGAR string by 10 bases containing the highest percentage of both shared SNPs, 96.41% and indels, 88.62%.

Table 4.4: **NCYC1026.** The proportion of shared SNPs and indels both before and after filtering NCYC1026 sequence reads derived from NCYC sequencing Plate 1 and the prior CCC project sequence set. The unfiltered variant calls were compared to variants filtered on the allelic fraction. The highest percentages of shared variants are highlighted in red.

Read	All Frequencies			AF < 0.125			0.125 <= AF < 0.875			AF >= 0.875		
Anchoring	P1	Both	CCC	P1	Both	CCC	P1	Both	CCC	P1	Both	CCC
SNPs												
unfiltered	4.98	91.81	3.21	6.03	3.26	90.72	4.64	77.97	17.40	2.97	92.70	4.33
CIGAR 10	5.14	91.80	3.06	7.00	3.93	89.07	4.74	82.81	12.45	2.86	92.09	5.05
CIGAR 20	5.21	91.82	2.97	7.19	3.98	88.84	4.77	82.76	12.46	2.91	92.06	5.03
CIGAR 30	5.32	91.72	2.96	7.34	4.34	88.32	4.98	82.47	12.55	2.95	92.02	5.03
FAT-CIGAR 10	2.12	**91.93**	5.95	17.36	11.14	71.50	4.03	**91.22**	4.75	2.16	94.25	3.59
FAT-CIGAR 20	1.09	88.09	10.82	25.23	33.24	41.53	6.20	87.44	6.35	1.82	95.26	2.92
FAT-CIGAR 30	0.51	76.46	23.03	26.14	**43.85**	30.00	10.24	79.75	10.02	1.97	**95.32**	2.71
Indels												
unfiltered	9.86	84.33	5.81	23.96	19.74	56.30	9.32	**81.11**	9.57	3.72	**87.93**	8.34
CIGAR 10	10.16	84.81	5.03	26.25	18.85	54.90	9.16	81.09	9.76	3.96	80.49	15.55
CIGAR 20	10.01	**86.06**	3.93	23.91	21.70	54.39	8.96	80.31	10.72	3.58	80.42	16.01
CIGAR 30	12.07	84.48	3.45	23.05	**29.55**	47.40	12.61	72.40	14.99	5.06	77.97	16.98
FAT-CIGAR 10	9.69	84.51	5.81	25.11	20.50	54.39	9.55	80.21	10.25	3.90	81.93	14.18
FAT-CIGAR 20	9.07	82.19	8.74	20.44	28.82	50.74	9.66	78.61	11.72	3.80	82.60	13.60
FAT-CIGAR 30	9.55	72.39	18.05	24.14	27.53	48.34	12.23	71.83	15.93	5.19	80.83	13.98

Table 4.4 shows the percentage of shared variants after filtering for the tetraploid strain NCYC1026. As with NCYC87, variants were filtered on the allelic fraction according to their expected frequencies based on the ploidy level. In this case, expected variant frequencies were 0.25, 0.5, 0.75 and 1.00. Reads filtered on the FAT-CIGAR string by 10 bases had a slightly higher percentage of shared SNPs, 91.93%, from the unfiltered calls whilst filtering on the CIGAR string by 20 bases produced the highest percentage of shared indels, 86.06%. Variant calls with AF $<$ 0.125 were also found to have the lowest proportion of shared variants whilst variants filtered with AF $>=$ 0.875 had the highest proportion of shared variants. For calls with AF $<$ 0.125, a greater increase in the proportion of shared SNPs was seen when filtered on the FAT-CIGAR string by 30 bases, 40.5%, whilst indels from reads filtered on the CIGAR string by the same length showed a 9.81% increase. For the heterozygous variants with AF between 0.125 and 0.875, filtering on the FAT-CIGAR string by 10 bases increased the

percentage of shared SNPs by 13.25% whilst it caused the percentage of shared indels to steadily decrease. This was also found to be the case for homozygous indels with AF >= 0.875 however, reads filtered on the FAT-CIGAR string had a greater proportion of shared variants than reads filtered on the CIGAR string for the corresponding anchor lengths. For SNP calls, filtering on the FAT-CIGAR string by 30 bases increased the percentage of shared SNPs by 2.62%.

4.4 Discussion

A novel method of read filtering using the *FAT-CIGAR* toolkit was explored in this study as a strategy to improve variant calling. The *FAT-CIGAR* toolkit was further developed to allow for the ends of sequence reads to be anchored against the reference genome by a specified number of bases using either the FAT-CIGAR or the CIGAR string. This method of filtering was implemented to combat high false positive discovery rates, particularly with regard to SNP calling. Spurious false positive SNP calls arise from either sequencing errors or the misalignment of reads spanning repetitive regions and structural variant breakpoints. Many of these SNPs are likely to have abundant read support preventing existing variant filtration techniques from identifying them as false positive SNPs. The *FAT-CIGAR* toolkit was designed to overcome this limitation by filtering out reads that contain variants at either end prior to variant calling.

A suitable base length on which to carry out read filtering was originally selected by observing the percentage of sequence reads retained within a BAM file after filtering. The read retention was monitored to help identify the cut-off value at which filtering became too stringent. A highly stringent filter could discard a huge proportion of the sequence reads resulting in several true positive variants being filtered out along with the false positive calls. The maximum length by which to anchor reads was determined to be 30 bases. *pIRS* simulates short sequence reads of length 100 bp by default. Thus, as both ends of the read have to be anchored against the reference, this requires the read to have a greater than 60% sequence identity score with the reference genome. In the first analysis, the reads were filtered on the FAT-CIGAR string as the CIGAR string masks SNPs, therefore it would not have precluded reads containing false positive SNPs from passing through. The percentage of reads retained was shown to steadily decrease as the filtering stringency increased. The overall read retention was found to

remain quite high, 78.7%, even when filtered at 30 bases. This can be attributed to the fact that sequence reads were simulated from the S288c chromosome I and re-mapped against the reference hence, the high sequence identity was to be expected. As this was the initial testing phase of the study, reads were purposefully simulated from the reference genome to ensure that the only source of variation within the reads was due to sequencing errors simulated by *pIRS*. This enabled us to test whether filtering on the FAT-CIGAR string can remove reads containing sequencing errors. Visualisation of the alignments within the filtered BAM files also helped confirm that read filtering was able to remove sequencing errors. In addition, reads that were partially aligned, thus a portion of the sequence is masked by either soft- or hard-clipped bases, were also filtered out.

As discussed in the previous chapter, it is critical to have a benchmark dataset against which the accuracy of variant calls can be evaluated. The pipeline developed for the generation of a truthset in Chapter 3 was highly complex and dependent on several third-party software to acquire the necessary input parameters. Consequently, despite the pipeline requiring significant modification, it did not allow for finer control of the variant simulation that was necessary for this study. There are several software for genome simulation such as *PGsim* (Juan *et al.*, 2020) and *VarSim* (Mu *et al.*, 2015) however they are tailored specifically for human genomes and require third-party databases as input. The *sim_genomes* program was developed to allow for simplicity and control over the genome simulation and to combine the truthset generation into a single step process.

The S288c chromosome I was chosen as the reference sequence from which genome simulation was carried out, in order to reduce the computational complexity of working with whole genomes. The *pIRS* in-house error profiling scripts were used to generate custom error profiles using NCYC78 as the standard error profile utilised in *pIRS* was obtained from large-scale human sequencing data. Prior to this, the 10 NCYC strains from the read mapping comparison study (see Chapter 1) were selected for profiling but the *baseCalling_Matrix_calculator* for generating substitution profiles could not be run on the merged SAM files. The indel profiling script for NCYC78 also produced an error due to a bug in a Perl script that could not be fixed, therefore the standard indel profile was utilised instead. These factors contributed to inappropriate read simulation when

using the custom error profile, leading to a much higher number of sequencing errors simulated within the reads than variants. The proportion of false positive variants was only found to decrease below the proportion of true positive variants when filtered at 20 bases, as this highly conservative filter resulted in 84% of the false positive variants being removed. These findings indicated that the custom error profile is not suitable for subsequent analyses, until these software errors are resolved. Variants classified by *vcfeval* were manually verified against the truthset, indicating that the software failed to identify variants represented differently after normalisation. The misclassified variants were subsequently error corrected to reduce the number of incorrect false positive and false negatives variants.

As hypothesised, filtering reads on the FAT-CIGAR string was found to improve SNP calling accuracy by greatly decreasing the number of false positive variant calls. The maximum anchor length for filtering was chosen to be 20 bases as this filter was found to be highly conservative, resulting in over 50% of the sequence reads to be discarded for the genome containing 3,000 SNPs. The stringency of the read filter also resulted in a greater number of true positive variant calls being filtered out as increasing the incidence of SNPs increased the likelihood of the SNPs occurring close to the read ends. Filtering at 10 bases was found to be optimal, removing over 60-70% of false positive variant calls without filtering out any true positives. Only 1-3 true positive SNPs were removed across the three genomes when filtered at 10 bases. This confirmed that the majority of the SNPs that are predicted close to read ends are false positives that occur due to sequencing errors or misalignment of reads. Therefore, the removal of these SNPs is crucial for accurate downstream analyses.

The *sim_genomes* program was developed to simulate indels, utilising a similar method to that used in the *SInC* software. The use of this method granted greater control over the number and size of the indels simulated and reduced the complexity of the simulation. Regions of the reference genome containing repeat expansions had to be identified using the *TRF* software prior to genome simulation. In the future, it may be useful to be able to run the *TRF* program in-house from the script using the provided reference. The overlapping regions could be compared to select for the repeat sequences with the highest alignment scores from within the program as the repeat sequences were chosen manually for this study. Ideally, with more time, it would

also have been beneficial to introduce large-scale structural variants including inversions and translocations. This would have enabled for an in-depth study of how read filtering impacts variant calling around regions containing large-scale structural variation.

Read filtering on the CIGAR string which masks SNPs but not indels was introduced to compare the accuracy of calling indels against reads filtered on the FAT-CIGAR string. Filtering on both the CIGAR and FAT-CIGAR strings was found to reduce the number of false positive SNPs and indels called by removing the reads containing false positive variants prior to variant calling. The reads filtered on the FAT-CIGAR string were able to call fewer false positive variants compared to reads filtered on the CIGAR string. This difference was mainly seen in the more numerous SNP calls as SNPs that would otherwise be removed when filtering on the FAT-CIGAR string are not accounted for when filtering on the CIGAR string. As some of the discarded reads are likely to have contained false positive indels, the removal of these reads also resulted in fewer false positive indel calls when filtering on the FAT-CIGAR string. This reduction in false positives calls was, however, found to be at the expense of also filtering out true positives, resulting in an increased number of false negatives as the stringency of filtering increased. This was especially true for SNP calls when anchored on the FAT-CIGAR string by more than 10 bases as demonstrated by the declining F1 scores. Filtering at 30 bases was found to be too conservative as requiring that level of exact sequence identity resulted in a high number of true variants being removed, thus decreasing the variant calling accuracy to below that of the unfiltered variant calls. For three of the four genomes, the most accurate indel calls were from reads filtered on the FAT-CIGAR string by 20 bases. The genome with the greatest number of indels (300) had fewer false positives and a higher F1 score when filtered on the CIGAR string by 30 bases. A likely explanation was that increasing the incidence of indels also increased the number of false positives that arise from misalignment of reads, therefore anchoring conservatively at 30 bases is able to discard a greater proportion of reads containing false positive indels.

Imitation of the variants within five strains from the Bergstrom strain population enabled a better understanding of how filtering reads could affect variant calling in genomes with realistic variant numbers. The strains were chosen from the five different sub-populations to ensure increased diversity in strain selection. Filtering on the CIGAR

string by 30 bases produced fewer false positives than filtering on the FAT-CIGAR string for the majority of the simulated genomes and resulted in more accurate variant calls as determined by the F1 score. This confirmed that the finding in the previous study was not due to dataset bias. It also supported the idea that choosing an appropriate read filter to improve the accuracy of indel variant calling was dependent on the level of sequence diversity within the genome. For indels, filtering on the FAT-CIGAR string by 20 bases was found to be optimal when there are fewer SNPs present. However, as the number of variants within the genome increases, this results in a higher coincidence of SNPs and indels occurring on the same sequence read. Hence, the removal of these reads when filtering on the FAT-CIGAR string by 30 bases to target false positive SNPs results in the removal of a huge proportion of true positive indels. By permitting these reads to remain, by filtering on the CIGAR string by 30 bases, allows a reduction in false positive indel calls due to the stringent anchor length whilst also retaining the true indels. Reads filtered on the FAT-CIGAR string by 10 bases were still found to prevail in the accuracy of SNP calling. The F1 scores across the genomes were found to be highly similar despite the difference in the number of variants due to the similarity in the ratio of SNPs to indels. The ratio of SNPs to indels were as follows: 1:0.06 for Y12, 1:0.08 for DBVPG6765, YPS128 and DBVPG6044 and 1:0.09 for DBVPG6044.

A number of genomes were simulated with varying ratios of SNPs and indels to re-affirm the results seen in the previous study regarding the optimal filter for variant calling. As is the case for most studies, the number of variants within the genome is not known until after variant calling. Thus, in the absence of this knowledge, the recommended read filter for SNP calling would be to filter on the FAT-CIGAR string by 10 bases as this produced the most consistently accurate variant calls. However, for genomes containing a greater ratio of indels, increasing the anchor length to 20 bases would be recommended. The SNP calling accuracy of genomes with a high SNP to indel ratio can be optimised by using a less conservative approach and filtering on the CIGAR string by either 20 or 30 bases. The recommended filter for accurate indel calling would be to filter on the CIGAR string by 30 bases. If variant calling on a genome that is known to have very low sequence diversity compared to the reference, a more stringent approach of filtering on the FAT-CIGAR string by 30 bases should be taken to optimise the accuracy of the indel calls.

The impact of read filtering on real genomes was investigated by observing the changes in the percentage of shared variants between the same strains that were sequenced on different plates. Filtering the sequence reads on the FAT-CIGAR string by either 10 or 20 bases for SNP calling and on the CIGAR string by 20 or 30 bases for indel calling was shown to increase the percentage of shared variants for the unfiltered calls. This was likely due to false positive variants being unique to the reads from a specific sequencing plate whereas true positive variants will be present in the reads from both plates. Filtering the reads should remove the false positive calls resulting in a greater proportion of true positive variants and therefore, increasing the percentage of shared variants. This was confirmed by the greatest increase in the percentage of shared variants being from reads filtered on variants with an allelic fraction < 0.16 which likely contained the greatest number of false positives. For NCYC87 and NCYC1026, the conservativeness of read filtering was not found to be necessary for indels with a high allelic fraction, particularly for the heterozygous variants. This suggested that the indel calls likely contained fewer false positive variants, thus filtering was removing a greater number of true positives causing the percentage of shared indels to decrease. These findings strongly indicated that read filtering also improves the accuracy of variant calling in real genomes.

As of yet, the *FAT-CIGAR* toolkit does not provide the option to utilise different filters for SNPs and indels. The necessity of using separate filters to optimise the accuracy of SNPs and indel calling has been firmly highlighted in this study. The recommended approach at present would be to carry out filtering by 10 bases on the FAT-CIGAR string and filter out the SNPs from the VCF file. Subsequently, filtering on the CIGAR string by 30 bases and combining the indels called with the prior SNPs would produce an optimal variant set. In future, it would be of great benefit to modify the read filtering function in the *FAT-CIGAR* toolkit so that different filters for SNPs and indels could be specified by the user. The variants within each read could be distinguished individually as to whether it is a SNP or indel and used to determine whether the read is filtered based on either the CIGAR or FAT-CIGAR string. The recommended filters could also be built-in as the default filtering option. This added functionality would play a pivotal role in truly optimising variant calling accuracy.

4.5 Conclusion

Conventional methods for variant filtration struggle to distinguish false positive variants that have high read depth, leading to inaccuracies in variant discovery. The *FAT-CIGAR* toolkit was found to be highly effective at removing false positive variants, prior to variant calling, by filtering reads on either the CIGAR or FAT-CIGAR string. An additional outcome of this study was the development of the *sim_genomes* program to simulate genomes without the complications of using several third-party software for the generation of a variant truthset. The accuracy of variant calling, as determined by F1 scores, across several simulated genomes was shown to improve greatly on the filtered reads. Generically, the optimal read filter for SNP calling was identified to be filtering on the FAT-CIGAR string by 10 bases whilst filtering on the CIGAR string by 30 bases produced accurate indel calls. The accuracy of the filter was also found to be dependent on the level of variation within the genome, thus prior knowledge of the genome can aid in the selection of a more appropriate read filter. Further work needs to be carried out on the *FAT-CIGAR* toolkit to allow for SNPs and indels to be filtered separately and the impact of read filtering on the accuracy of calling structural variants should also be investigated.

Chapter 5

Discussion

5.1 Main Goals

This project fundamentally set out to establish optimised strategies for studying genetic variation within yeast genomes, particularly *Saccharomyces cerevisiae*. The main goal of this project was to identify a functional pipeline that utilises both novel and existing methods to improve the accuracy of variant discovery using next-generation sequencing datasets. For the purpose of this study, improved accuracy was primarily defined by an increase in the number of true variants called and a reduction in false positive variant calls. This goal was predominantly pursued by adopting the use of pan-genome variation graphs as reference structures to refine the assembly of short sequence reads. A key objective of this study involved identifying a viable measure for carrying out cross-comparison of sequence read alignment quality across different software. Another objective was to investigate the performance of the graph genome software, in terms of accuracy of the variant calls, against the conventional variant calling pipeline. The final aim of this project to optimise variant discovery was to explore novel techniques of read alignment filtration that greatly reduce the occurrence of false positive variant calls.

5.2 Outcomes

5.2.1 Variation Graphs

In Chapter 2, visualisation of variation graphs constructed using the *vg* toolkit helped gain understanding of how individual genomes are represented within the graph. The

node to sequence length ratio showed that the mitochondrial genome displayed the greatest degree of intragenic variation followed by chromosome I, whilst both chromosomes XIII and III were found to be highly conserved.

The Bergstrom strains were utilised to construct a variation graph to compare the efficacy of read mapping in comparison to the S288c reference graph and the linear reference genome. The variation graph had the greatest proportion of reads aligned for all 19 strains. This confirmed that prior inclusion of variants within the reference reduced bias in mapping and increased the ability of sequence reads to align. The reference graph also outperformed the linear reference genome which suggested that the *vg* mapping algorithm had greater sensitivity than *BWA*. The mitochondrial genome contains at least 15% of a strain's total DNA content yet only 2-5% fewer reads were able to align against the reference graph. This supported that a high proportion of mitochondrial-derived sequence reads still remain unaligned due to the greater degree of sequence diversity. The huge variance in the raw sequence coverage per strain and the overall proportion of reads that aligned, 38-91%, indicated poor sequence read quality for some datasets. Quality control analysis using *FastQC* highlighted the presence of adapter contamination within the reads that were subsequently removed through trimming. Trimming discarded a large proportion of the sequence reads for the majority of the Bergstrom strains but improved read alignment, with the average percentage of mapped reads increasing by 22.3% for the variation graph, 22.8% for the reference graph and 26.8% for the linear reference genome. The proportion of mapped reads was also found to increase across the references, 88-98%, when using high quality sequence reads from 10 NCYC strains. As both NCYC78 and NCYC97 had high sequence read quality, trimming the strains removed reads that would otherwise have aligned against the graph-based references, resulting in a decrease in mapped read counts.

Alignment scores were used to gain insight into the quality of alignments against each reference structure by ensuring similar scoring systems were employed by *vg* and *BWA*. However, manual inspection of the alignments identified several hidden differences in the scoring parameters that biased the scores in favour of *vg*. In addition, the alignment scores for certain reads in *BWA* were not representative of the final alignment thus rendering alignment scores redundant as a measure of mapping quality.

Reference genomes and sequence reads of varying lengths were simulated with different types and number of mutations to identify factors that prevent mapping. Variants that appeared towards read ends did not impact mapping as these sequences were masked through base clipping. At shorter read lengths, sequence reads were found to be less tolerant to variants as it prevented read alignment against the reference. Of all variant types, point mutations were found to have the greatest impact on read mapping ability with decreasing read length causing a rapid decline in mapping. As sequence reads containing point mutations are representative of highly divergent strains, this demonstrated their inability to align against the reference genome, reinforcing the need for variation graphs as reference structures.

5.2.2 Graph Genome Software

One of the main aims of this study was to compare the performance of read mapping and variant calling algorithms from recently released graph genome software against the conventional linear pipeline, as discussed in Chapter 3. Alignments could be obtained in BAM format for *BWA*, *vg* and the *Seven Bridges Graph Genome* toolkit thus the accuracy of read mapping across the three software was evaluated by BAM file comparison. The *FAT-CIGAR* toolkit was developed to obtain the FAT-CIGAR string, an extension of the CIGAR string, that provides the exact alignment information of a sequence read to a reference structure. This was utilised to calculate accurate sequence identity scores to be used as a measure of the alignment quality. For linear alignments, the toolkit used the MD tag information, which distinguished between matched and mismatched bases within the alignment, to create the FAT-CIGAR string. The global alignment score could also be calculated for such sequence read and the CS tag, another alignment representation format, could be generated to enhance the functionality of the toolkit. The *FAT-CIGAR* toolkit constructed the FAT-CIGAR string from the non-surjected alignments for *vg* using information within the output JSON file. The toolkit generated the non-surjected CIGAR string for alignments from the *Graph Genome* toolkit.

A greater proportion of reads were able to align to the *vg* variation and reference graphs than to the *Graph Genome* toolkit, which reaffirmed that the *vg* mapping algorithm had greater sensitivity. The variation graph was found to improve the quality of mapping for both software by increasing the number of reads with greater sequence

identity scores due to the decreased ambiguity in mapping. The *Graph Genome* toolkit was found to align more reads with greater overall accuracy whilst *vg* had the greatest percentage of perfectly mapped reads. Although fewer reads were able to align against the linear reference genome, the quality of alignment was found to be slightly better than that of alignments against the reference graph.

The accuracies of the variant calls produced by *vg*, the *Graph Genome* toolkit, *BayesTyper* and *Graphtyper* were also compared against the *FreeBayes* variant calls. An increased number of SNPs and indels were called against the variation graph in comparison to the reference graph for all software. The *vg* variation graph called the highest number of SNPs for each strain which indicated the likelihood of a greater number of false positive variant calls in *vg*. A simulation study was conducted to simulate 1,000 datasets, each containing 19 genomes, from the Bergstrom strains SNP tree and to generate truth sets to compare variant calling accuracy. Generating the pipeline for the simulation study proved to be a highly challenging process that required several modifications to ensure the simulated genomes mimicked the real strains. Variant classification with *vcfeval* was identified to be error-prone thus falsely classified variants had to be corrected. Preliminary analysis of the variant calls determined *vg* v1.26 to be unstable as it contained a vastly greater number of false positive calls thus *vg* v1.18 and v1.5, which were utilised in the earlier chapters, were also re-run to test for stability. All three versions of *vg* produced the least accurate variant calls, as determined by F1 scores, due to calling fewer true positives and a greater number of false positives. This strongly suggested that the *vg* variant calling algorithm was inherently less accurate than alternative approaches. Variant calls from the variation graph were found to be more accurate due to containing more true positive calls than the reference graph. The recommended software for variant calling was found to be the *Graph Genome* toolkit which had the most accurate variant calls with the greatest sensitivity for all strains, whilst *Graphtyper* had the greatest precision.

5.2.3 Variant Filtration

In Chapter 4, the functionality of the *FAT-CIGAR* toolkit was further extended to anchor both ends of qualifying sequence reads against the reference genome using either the CIGAR or the FAT-CIGAR string, by a specified number of bases. This method of

variant filtration was hypothesised to remove false positive calls that are not removed during hard filtering due to immense read support. In order to avoid the complexity of the simulation pipeline in the previous study, the *sim_genomes* program was implemented to simulate genomes with a specified number of SNPs and indels. The variants induced within the simulated genome were monitored and written out as the benchmark VCF file utilised to classify the variant calls.

A simulation study was conducted by simulating a chosen number of genomes with varying proportions of SNPs and indels to identify the read filter that produced the most optimal variant calls. The accuracy of SNP and indel calls were examined individually to investigate how the choice of string impacts variant calling. The accuracy of SNP calling was found to increase when reads were filtered on the FAT-CIGAR string as the approach was able to distinguish mismatched bases that can often give rise to false positive SNPs from matched bases in the anchor region. Anchoring reads on the FAT-CIGAR string by 10 bases filtered out false positive SNPs whilst preserving true positive SNP calls. Genomes containing a greater SNP to indel ratio required reduced stringency by filtering on the CIGAR string by either 20 or 30 bases for optimal SNP calling. Reads filtered on the CIGAR string by 30 bases produced the most accurate indel calls. Filtering on the CIGAR string prevented a greater number of reads being discarded whilst the conservative nature of the anchor length allowed for false positive indels to be removed. If the genome contains low sequence diversity, filtering stringently on the FAT-CIGAR string by 30 bases enabled a reduction in false positive indel calls.

The proportion of shared variants between sequence reads from strains that were sequenced on different sequencing plates were observed after filtering. Filtering the reads on the FAT-CIGAR string for SNPs and CIGAR string for indels increased the proportion of shared variants, which indicated that filtering out false positive calls increased the percentage of true positive variants. These findings strongly advocate for read filtering using the FAT-CIGAR software to be utilised as part of routine variant calling pipelines.

5.3 Future Directions

5.3.1 Variation Graphs

The Bergstrom strain set could be re-sequenced to obtain good quality sequence reads as the poor read quality had a considerable impact on read alignment. The DNA libraries could be normalised during sequencing preparation to ensure equal read coverage across the strains. The node to sequence length ratio provided an indication of the intragenic diversity within the Bergstrom strain population, however the degree of conservation seen within each chromosome was not further explored in this study. Only the percentage of reads that were able to align against the mitochondrial genome in the reference graph was investigated. This study could have been extended to observe the percentage of reads that were able to align against each individual chromosome for the variation graph, reference graph and linear reference genome. This could have enabled direct quantification of the correlation between the degree of diversity and the ability of reads to align against each reference structure.

Both the NCYC and Bergstrom strains utilised for the construction of the variation graphs in this study were haploid, thus the impact of mapping strains with varying levels of ploidy was not explored beyond a small analysis using the tetraploid strain NCYC1006. A subset of diploid, triploid and tetraploid *S. cerevisiae* strains could have been selected from the NCYC collection and aligned against the three reference structures. As strains with varying ploidy levels contain greater genome content variation, this could have allowed for an in-depth investigation of how read mapping is impacted in the presence of pervasive large-scale structural variation.

Large-scale sequencing studies on *S. cerevisiae* have resulted in publicly available whole-genome sequences for thousands of strains (Peter *et al.*, 2018, Zhu *et al.*, 2016). A pan-genome variation graph could have been constructed as the optimal reference structure for *S. cerevisiae* through the careful selection of a subset of strains from the collection of genomes. K-mer profiling could be carried out using software such as Jellyfish (Marcais and Kingsford, 2011) prior to conducting phylogenetic analysis to identify strains from diverging populations. This could immensely improve computational efficiency and reduce data redundancy in graph construction.

The simulation study focused on factors that impact mapping in *vg* using the refer-

ence graph. This could have been extended to also evaluate how variants impact read mapping in *BWA* and how read tolerance to variants increases against the variation graph.

5.3.2 Graph Genome Software

The run time for the *FAT-CIGAR* toolkit for alignments from *vg* was found to increase exponentially as the number of sequence reads increased, due to having to store the JSON file into memory. The run time could be reduced by adding functionality into the toolkit that breaks up the GAM files into individual chromosome files and multi-threading could be used to run the files through the script in parallel. Reducing the file size could reduce the memory usage of the script, speeding up the run time considerably.

The alignment comparison study utilised *vg* version 1.26 which was the latest version at the time the study was conducted. The software is still under rapid development thus several further updates have been released since, with version 1.34 being the current version. This version contains major changes to the graph construction, indexing and mapping algorithms. The alignment comparison study could be re-run using the latest version of *vg* to evaluate whether recent changes to the algorithm impacted on the accuracy of mapping.

The study could also have focused on differences in the accuracy of alignments spanning regions of large-scale structural variation for each reference structure. A few regions of structural variation could have been simulated and the alignment quality of reads that mapped to these regions could have been compared based on sequence identity scores. This could enable an understanding of whether the use of variation graphs improves both the alignment and the accuracy of mapping within these regions.

As *vg* is able to align PacBio sequence reads against the variation graph, the difference in mapping accuracy when using long reads versus short shorts could also have been studied. The differences in the proportions of shared variants across each reference structure for variant calls from the Bergstrom strains could also have been compared. The variant calling algorithm in *BayesTyper* was determined to be non-deterministic, therefore a consensus approach could have been taken in which each dataset was run at

least 10 times. Obtaining an average for the number of variants called across multiple runs could have provided a better indication of variant calling accuracy. Evaluating the accuracy of SNP and indel calls separately could have provided better insight into the performance of each graph genome software. The observed differences in accuracy could be used to understand whether variation graphs have a greater impact on SNP or indel calling. A few datasets could have been simulated from the whole genome to compare whether the accuracy of the variant calls remained constant across different chromosomes.

5.3.3 Variant Filtration

The *sim_genomes* program heavily relies on the user to supply the DAT file containing the curated repeat expansions. The program could be further developed so that the *TRF* repeat finding software is run from within the program such that repeat sequences are automatically selected based on their alignment scores. This could ensure that the DAT file does not contain repeat sequences from overlapping regions. The *sim_genomes* program is able to simulate large-scale indels and copy number variation, but its functionality could be further improved by the generation of complex structural variants such as inversions and translocations.

Default error profiles within the *pIRS* software, that were generated from human genomes, were utilised for read simulation as the custom error profiles could not be generated. Read simulation could be carried out using the *ART* read simulator which could allow for error profiles to be generated directly from the NCYC strain sequence reads. The accuracy of SNP and indel calling could have been observed particularly around the structural variant regions and breakpoints. This could permit understanding of whether filtering reduces false positive variant calls within these regions and if greater stringency in filtering is required compared to the rest of the genome.

The impact of read filtering on the proportion of shared variants was observed only for *S. cerevisiae* genomes. This study could have been extended to incorporate strains from other yeast species within the Saccharomyces clade that had been re-sequenced, such as *Saccharomyces bayanus* or potentially strains from a different genus.

The *FAT-CIGAR* toolkit could be modified to allow the user to specify separate filters for SNPs and indels. The default option of filtering SNPs on the FAT-CIGAR string and indels on the CIGAR string could be built into the software.

5.4 Final Conclusion

The process of identifying true sequence variants from within a genome is highly challenging and requires the utmost accuracy in every step of the pipeline. Many of the limitations faced by the current approach to variant discovery have been clearly highlighted in this study. The importance of certain aspects of the pipeline has also been emphasised, such as carrying out quality control analysis to determine the dataset quality prior to read mapping.

The constraints of mapping to a linear reference genome were mitigated by the use of variation graphs as yeast reference structures. Despite the improvement in alignment quality only translating to improved accuracy in variant calling for the *Seven Bridges Graph Genome* toolkit, many of the software were found to have near perfect precision in variant calling. The relative novelty of the field of graph genomics means that the software utilised in this study are still undergoing rapid development. Thus, the release of further graph genome software and improved accuracy in the variant calling algorithms of the current software can be expected with time. Hard filtering is usually carried out post variant calling and is critical for the removal of false positive variant calls. The concept of variant filtration, through the anchoring of sequence reads on either the CIGAR or FAT-CIGAR string, utilised in this study was also found to be highly effective in the removal of false positives. The adoption of a combination of methods outlined in this study will allow for truly optimised, accurate variant discovery.

Bibliography

[1] 1000 Genomes Project Consortium. A map of human genome variation from population scale sequencing. *Nature 467* (2010), 1061–1073.

[2] Acinas, S. G., Sarma-Rupavtarm, R., Klepac-Ceraj, V., and Polz, M. F. PCR-induced sequence artifacts and bias: insights from comparison of two 16S rRNA clone libraries constructed from the same sample. *Applied and Environmental Microbiology 71*, 12 (2005), 8966–8969.

[3] Ameur, A. Goodbye reference, hello genome graphs. *Nature Biotechnology 37*, 8 (2019), 866–868.

[4] Andrews, S. A quality control tool for high throughput sequence data. `http://www.bioinformatics.babraham.ac.uk/projects/fastqc/`, 2010.

[5] Audano, P. A., Sulovari, A., Graves-Lindsay, T. A., Cantsilieris, S., Sorensen, M., Welch, A. E., Dougherty, M. L., Nelson, B. J., Shah, A., Dutcher, S. K., et al. Characterizing the major structural variant alleles of the human genome. *Cell 176*, 3 (2019), 663–675.

[6] Azhar, S. H. M., Abdulla, R., Jambo, S. A., Marbawi, H., Gansau, J. A., Faik, A. A. M., and Rodrigues, K. F. Yeasts in sustainable bioethanol production: A review. *Biochemistry and Biophysics Reports 10* (2017), 52–61.

[7] Bankar, A. V., and Kumar, Ameeta R.and Zinjarde, S. S. Environmental and industrial applications of *Yarrowia lipolytica*. *Applied Microbiology and Biotechnology 84*, 5 (2009), 847–865.

[8] Benjamini, Y., and Speed, T. P. Summarizing and correcting the GC content bias in high-throughput sequencing. *Nucleic Acids Research 40*, 10 (2012), e72–e72.

[9] Benson, G. Tandem repeats finder: a program to analyze DNA sequences. *Nucleic Acids Research 27*, 2 (1999), 573–580.

[10] Bergström, A., Simpson, J. T., Salinas, F., Barré, B., Parts, L., Zia, A., Ba, A. N. N., Moses, A. M., Louis, E. J., Mustonen, V., and et al. A high-definition view of functional genetic variation from natural yeast genomes. *Molecular Biology and Evolution 31*, 4 (2014), 872–888.

[11] Bolger, A. M., Lohse, M., and Usadel, B. Trimmomatic: a flexible trimmer for Illumina sequence data. *Bioinformatics 30*, 15 (2014), 2114–2120.

[12] Brown, J., Pirrung, M., and McCue, L. A. FQC Dashboard: integrates FastQC results into a web-based, interactive, and extensible FASTQ quality control tool. *Bioinformatics 33*, 19 (2017), 3137–3139.

[13] Bult, C. J., White, O., Olsen, G. J., Zhou, L., Fleischmann, R. D., Sutton, G. G., Blake, J. A., FitzGerald, L. M., Clayton, R. A., Gocayne, J. D., et al. Complete genome sequence of the methanogenic archaeon, *Methanococcus jannaschii*. *Science* (1996), 1058–1073.

[14] Burrows, M., and Wheeler, D. J. A block-sorting lossless data compression algorithm. `http://www.hpl.hp.com/techreports/Compaq-DEC/SRC-RR-124.html`, May 1994. Accessed: 2018-03-19.

[15] Bushnell, Brian. BBMap. `https://sourceforge.net/projects/bbmap/`, 2014.

[16] Cao, J., Schneeberger, K., Ossowski, S., Günther, T., Bender, S., Fitz, J., Koenig, D., Lanz, C., Stegle, O., Lippert, C., Wang, X., Ott, F., Müller, J., Alonso-Blanco, C., Borgwardt, K., Schmid, K. J., and Weigel, D. Whole-genome sequencing of multiple *Arabidopsis thaliana* populations. *Nature Genetics 43*, 10 (2011), 956–963.

[17] Carter, D. *Saccharomyces* Genome Resequencing Project:User Manual, 2008. Accessed: 2019-12.

[18] Chen, X., Schulz-Trieglaff, O., Shaw, R., Barnes, B., Schlesinger, F., Källberg, M., Cox, A. J., Kruglyak, S., and Saunders, C. T. Manta: rapid detection of structural variants and indels for germline and cancer sequencing applications. *Bioinformatics 32*, 8 (2016), 1220–1222.

[19] Chidgeavadze, Z., Beabealashvilli, R., Atrazhev, A., Kukhanova, M., Azhayev, A., and Krayevsky, A. 2', 3'-Dideoxy-3'aminonucleoside 5'-triphosphates are the terminators of DNA synthesis catalyzed by DNA polymerases. *Nucleic Acids Research 12*, 3 (1984), 1671–1686.

[20] Chow, Y.-L., and Schwartz, R. The N-Best Algorithm: An Efficient Procedure for Finding the Top N Sentence Hypotheses. In *Proceedings of the Workshop on Speech and Natural Language* (1989), HLT '89, Association for Computational Linguistics, pp. 199–202.

[21] Church, D. M., Schneider, V. A., Steinberg, K. M., Schatz, M. C., Quinlan, A. R., Chin, C.-S., Kitts, P. A., Aken, B., Marth, G. T., Hoffman, M. M., Herrero, J., Mendoza, M. L. Z., Durbin, R., and Flicek, P. Extending reference assembly models. *Genome Biology 16*, 1 (2015), 1–5.

[22] Cleary, J. G., Braithwaite, R., Gaastra, K., Hilbush, B. S., Inglis, S., Irvine, S. A., Jackson, A., Littin, R., Rathod, M., Ware, D., Zook, J. M., Trigg, L., and De La Vega, F. M. Comparing variant call files for performance benchmarking of next-generation sequencing variant calling pipelines. *bioRxiv* (2015).

[23] Consortium, E. P. An integrated encyclopedia of DNA elements in the human genome. *Nature 489*, 7414 (2012), 57.

[24] Consortium, G. Genetic effects on gene expression across human tissues. *Nature 550* (2017), 204–213.

[25] Consortium, T. I. H. A second generation human haplotype map of over 3.1 million SNPs. *Nature 449*, 7164 (2007), 851–861.

[26] Danecek, P., Auton, A., Abecasis, G., Albers, C. A., Banks, E., Depristo, M. A., Handsaker, R. E., Lunter, G., Marth, G. T., Sherry, S. T., and et al. The variant call format and VCFtools. *Bioinformatics 27*, 15 (2011), 2156–2158.

[27] Darling, A. E., Mau, B., and Perna, N. T. progressiveMauve: Multiple Genome Alignment with Gene Gain, Loss and Rearrangement. *PLoS ONE 5*, 6 (06 2010), 1–17.

[28] Darriba, D., Taboada, G. L., Doallo, R., and Posada, D. jModelTest 2: more models, new heuristics and parallel computing. *Nature Methods 9*, 8 (2012), 772–772.

[29] Davey, R. P., James, S. A., Dicks, J., and Roberts, I. N. TURNIP: tracking unresolved nucleotide polymorphisms in large hard-to-assemble regions of repetitive DNA sequence. *Bioinformatics 26*, 22 (2010), 2908–2909.

[30] David, M., Dzamba, M., Lister, D., Ilie, L., and Brudno, M. SHRiMP2: sensitive yet practical short read mapping. *Bioinformatics 27*, 7 (2011), 1011–1012.

[31] De Bruijn, N. G. A combinatorial problem. In *Proc. Koninklijke Nederlandse Academie van Wetenschappen* (1946), vol. 49, pp. 758–764.

[32] Degner, J. F., Marioni, J. C., Pai, A. A., Pickrell, J. K., Nkadori, E., Gilad, Y., and Pritchard, J. K. Effect of read-mapping biases on detecting allele-specific expression from RNA-sequencing data. *Bioinformatics 25*, 24 (2009), 3207–3212.

[33] Del Fabbro, C., Scalabrin, S., Morgante, M., and Giorgi, F. M. An extensive evaluation of read trimming effects on Illumina NGS data analysis. *PLoS ONE 8*, 12 (2013), e85024.

[34] Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological) 39*, 1 (1977), 1–22.

[35] DePristo, M. A., Banks, E., Poplin, R., Garimella, K. V., Maguire, J. R., Hartl, C., Philippakis, A. A., Del Angel, G., Rivas, M. A., Hanna, M., et al. A framework for variation discovery and genotyping using next-generation DNA sequencing data. *Nature Genetics 43*, 5 (2011), 491–498.

[36] Dlamini, Z., Francies, F. Z., Hull, R., and Marima, R. Artificial intelligence (AI) and big data in cancer and precision oncology. *Computational and Structural Biotechnology Journal* (2020).

[37] Dolan, S. JQ. `https://github.com/stedolan/jq`, 2014.

[38] Dündar, F., Skrabanek, L., and Zumbo, P. Introduction to differential gene expression analysis using RNA-seq. Applied Bioinformatics Core/Weill Cornell Medical College, 2015.

[39] Durbin, R., Eddy, S. R., Krogh, A., and Mitchison, G. *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge University Press, 1998.

[40] Eberle, M. A., Fritzilas, E., Krusche, P., Källberg, M., Moore, B. L., Bekritsky, M. A., Iqbal, Z., Chuang, H.-Y., Humphray, S. J., Halpern, A. L., and et al. A reference data set of 5.4 million phased human variants validated by genetic inheritance from sequencing a three-generation 17-member pedigree. *Genome Research 27*, 1 (Jan 2017), 157–164.

[41] Edmonds, J., and Johnson, E. L. Matching: A well-solved class of integer linear programs. In *Combinatorial Optimization—Eureka, You Shrink!* Springer, 2003, pp. 27–30.

[42] Eggertsson, H. P., Jonsson, H., Kristmundsdottir, S., Hjartarson, E., Kehr, B., Masson, G., Zink, F., Hjorleifsson, K. E., Jonasdottir, A., Jonasdottir, A., Jonsdottir, I., Gudbjartsson, D. F., Melsted, P., Stefansson, K., and Halldorsson, B. V. Graphtyper enables population-scale genotyping using pangenome graphs. *Nature Genetics 49*, 11 (2017), 1654–1660.

[43] Engel, S. R., Dietrich, F. S., Fisk, D. G., Binkley, G., Balakrishnan, R., Costanzo, M. C., Dwight, S. S., Hitz, B. C., Karra, K., Nash, R. S., Weng, S., Wong, E. D., Lloyd, P., Skrzypek, M. S., Miyasato, S. R., Simison, M., and Cherry, J. M. The reference genome sequence of Saccharomyces cerevisiae: then and now. *G3: Genes, Genomes, Genetics 4*, 3 (2014), 389–398.

[44] Ewing, B., Hillier, L., Wendl, M. C., and Green, P. Base-calling of automated sequencer traces usingPhred. I. Accuracy assessment. *Genome Research 8*, 3 (1998), 175–185.

[45] Fawcett, J. A., Iida, T., Takuno, S., Sugino, R. P., Kado, T., Kugou, K., Mura, S., Kobayashi, T., Ohta, K., Nakayama, J.-i., and Innan, H.

Population genomics of the fission yeast *Schizosaccharomyces pombe*. *PLoS ONE 9*, 8 (08 2014), 1–12.

[46] Felsenstein, J. PHYLIP: phylogeny inference package. Version 3.2. *Cladistics 5* (1989), 164–166.

[47] Felsenstein, J., and Churchill, G. A. A Hidden Markov Model approach to variation among sites in rate of evolution. *Molecular Biology and Evolution 13*, 1 (1996), 93–104.

[48] Ferragina, P., and Manzini, G. Opportunistic data structures with applications. In *Proceedings 41st Annual Symposium on Foundations of Computer Science* (2000), pp. 390–398.

[49] Fleischmann, R., Adams, M., White, O., Clayton, R., Kirkness, E., Kerlavage, A., Bult, C., Tomb, J., Dougherty, B., Merrick, J., and al., e. Whole-genome random sequencing and assembly of *Haemophilus influenzae Rd*. *Science 269*, 5223 (1995), 496–512.

[50] Fletcher, W., and Yang, Z. INDELible: a flexible simulator of biological sequence evolution. *Molecular Biology and Evolution 26*, 8 (05 2009), 1879–1888.

[51] Fonseca, N. A., Rung, J., Brazma, A., and Marioni, J. C. Tools for mapping high-throughput sequencing data. *Bioinformatics 28*, 24 (2012), 3169–3177.

[52] Foury, F., Hu, J., and Vanderstraeten, S. Mitochondrial DNA mutators. *Cellular and Molecular Life Sciences CMLS 61*, 22 (2004), 2799–2811.

[53] Friedrich, A., Jung, P., Reisser, C., Fischer, G., and Schacherer, J. Population genomics reveals chromosome-scale heterogeneous evolution in a protoploid yeast. *Molecular Biology and Evolution 32*, 1 (2015), 184–192.

[54] Fuller, C. W., Middendorf, L. R., Benner, S. A., Church, G. M., Harris, T., Huang, X., Jovanovich, S. B., Nelson, J. R., Schloss, J. A., Schwartz, D. C., et al. The challenges of sequencing by synthesis. *Nature Biotechnology 27*, 11 (2009), 1013–1023.

[55] Garrison, E., Kronenberg, Z. N., Dawson, E. T., Pedersen, B. S., and Prins, P. Vcflib and tools for processing the VCF variant call format. *bioRxiv* (2021).

[56] Garrison, E., and Marth, G. Haplotype-based variant detection from short-read sequencing. *arXiv preprint* (2012).

[57] Garrison, E., Sirén, J., Novak, A. M., Hickey, G., Eizenga, J. M., Dawson, E. T., Jones, W., Garg, S., Markello, C., Lin, M. F., Paten, B., and Durbin, R. Variation graph toolkit improves read mapping by representing genetic variation in the reference. *Nature Biotechnology 36* (2018), 875.

[58] Garrison, E. A C++ library for parsing and manipulating VCF files. `https://github.com/vcflib/vcflib`, 2016.

[59] GATK. Variant Quality Score Recalibration (VQSR). `https://gatk.broadinstitute.org/hc/en-us/articles/360035531612-Variant-Quality-Score-Recalibration-VQSR-`, 2021. Accessed: 2021-08-15.

[60] Goddard, M. R., and Burt, A. Recurrent invasion and extinction of a selfish gene. *Proceedings of the National Academy of Sciences 96*, 24 (1999), 13880–13885.

[61] Goel, M., Sun, H., Jiao, W.-B., and Schneeberger, K. SyRI: finding genomic rearrangements and local sequence differences from whole-genome assemblies. *Genome biology 20*, 1 (2019), 1–13.

[62] Goffeau, A., Barrell, B. G., Bussey, H., Davis, R. W., Dujon, B., Feldmann, H., Galibert, F., Hoheisel, J. D., Jacq, C., Johnston, M., Louis, E. J., Mewes, H. W., Murakami, Y., Philippsen, P., Tettelin, H., and Oliver, S. G. Life with 6000 genes. *Science 274*, 5287 (1996), 546–567.

[63] Gotoh, O. An improved algorithm for matching biological sequences. *Journal of Molecular Biology 162*, 3 (1982), 705–708.

[64] Groth, C., Petersen, R. F., and Piškur, J. Diversity in organization and the origin of gene orders in the mitochondrial DNA molecules of the genus *Saccharomyces*. *Molecular Biology and Evolution 17*, 12 (2000), 1833–1841.

[65] Guo, Y., Li, J., Li, C.-I., Long, J., Samuels, D. C., and Shyr, Y. The effect of strand bias in Illumina short-read sequencing data. *BMC Genomics 13*, 1 (2012), 666.

[66] Hansey, C. N., Vaillancourt, B., Sekhon, R. S., de Leon, N., Kaeppler, S. M., and Buell, C. R. Maize (*Zea mays L.*) genome diversity as revealed by RNA-sequencing. *PLoS ONE 7*, 3 (2012), 1–10.

[67] Hasegawa, M., Kishino, H., and Yano, T.-a. Dating of the human-ape splitting by a molecular clock of mitochondrial DNA. *Journal of Molecular Evolution 22*, 2 (1985), 160–174.

[68] Huang, W., Li, L., Myers, J. R., and Marth, G. T. ART: a next-generation sequencing read simulator. *Bioinformatics 28*, 4 (12 2011), 593–594.

[69] Hyyrö, H. Bit-parallel approximate string matching algorithms with transposition. *Journal of Discrete Algorithms 3*, 2-4 (2005), 215–229. Combinatorial Pattern Matching (CPM) Special Issue.

[70] Ilie, L., and Ilie, S. Multiple spaced seeds for homology search. *Bioinformatics 23*, 22 (2007), 2969–2977.

[71] Illumina. An introduction to next-generation sequencing technology. `https://www.illumina.com/content/dam/illumina-marketing/documents/products/illumina_sequencing_introduction.pdf`, 2016. Accessed: 2018-03-19.

[72] Illumina. Illumina Adapter Sequences Document. `https://support.illumina.com/downloads/illumina-adapter-sequences-document-1000000002694.html`, 2019. Accessed: 2018-03-19.

[73] International Human Genome Sequencing Consortium. Finishing the euchromatic sequence of the human genome. *Nature 431*, 7011 (2004), 931–45.

[74] Iqbal, Z., Caccamo, M., Turner, I., Flicek, P., and McVean, G. De novo assembly and genotyping of variants using colored de Bruijn graphs. *Nature Genetics 44*, 2 (2012), 226–232.

[75] Jackson, P., and Joseph, S. Population genomics of yeasts: towards a comprehensive view across a broad evolutionary scale. *Yeast 33*, 3 (2016), 73–81.

[76] Jain, Sumit. Graph Representation – Adjacency Matrix and Adjacency List. `https://algorithms.tutorialhorizon.com/graph-representation-adjacency-matrix-and-adjacency-list/`, 2015. Accessed: 2019-05-03.

[77] Jones, W. Yeast Pan-Genomes: A Comparison Between Modern and Traditional Approaches, 2016.

[78] Juan, L., Wang, Y., Jiang, J., Yang, Q., Jiang, Q., and Wang, Y. PGsim: a comprehensive and highly customizable personal genome simulator. *Frontiers in Bioengineering and Biotechnology 8* (2020), 28.

[79] Junier, T., and Zdobnov, E. M. The Newick utilities: high-throughput phylogenetic tree processing in the UNIX shell. *Bioinformatics 26*, 13 (05 2010), 1669–1670.

[80] Kim, B.-Y., Park, J. H., Jo, H.-Y., Koo, S. K., and Park, M.-H. Optimized detection of insertions/deletions (INDELs) in whole-exome sequencing data. *PLoS ONE 12*, 8 (2017).

[81] Kokot, M., Długosz, M., and Deorowicz, S. KMC 3: counting and manipulating k-mer statistics. *Bioinformatics 33*, 17 (2017), 2759–2761.

[82] Kozarewa, I., Ning, Z., Quail, M. A., Sanders, M. J., Berriman, M., and Turner, D. J. Amplification-free Illumina sequencing-library preparation facilitates improved mapping and assembly of (G+ C)-biased genomes, Apr 2009.

[83] Krusche, P., Trigg, L., Boutros, P. C., Mason, C. E., De La Vega, F. M., Moore, B. L., Gonzalez-Porta, M., Eberle, M. A., Tezak, Z., Lababidi, S., Truty, R., Asimenos, G., Funke, B., Fleharty, M., Chapman, B. A., Salit, M., and Zook, J. M. Best practices for benchmarking germline small-variant calls in human genomes. *Nature Biotechnology 37* (2019), 555–560.

[84] Kumar, S., Stecher, G., Li, M., Knyaz, C., and Tamura, K. MEGA X: molecular evolutionary genetics analysis across computing platforms. *Molecular Biology and Evolution 35*, 6 (05 2018), 1547–1549.

[85] Kurtz, S., Phillippy, A., Delcher, A. L., Smoot, M., Shumway, M., Antonescu, C., and Salzberg, S. L. Versatile and open software for comparing large genomes. *Genome Biology 5*, 2 (2004), R12.

[86] Kurtzman, C. P., and Fell, J. W. Yeast systematics and phylogeny—implications of molecular identification methods for studies in ecology. In *Biodiversity and Ecophysiology of Yeasts*. Springer, 2006, pp. 11–30.

[87] Langmead, B. Introduction to the Burrows-Wheeler transform and FM index. http://www.cs.jhu.edu/~langmea/resources/bwt_fm.pdf. Accessed: 2018-03-19.

[88] Leinonen, R., Sugawara, H., and Shumway, M. The sequence read archive. *Nucleic Acids Res.*, D19-D21 (2011).

[89] Levenshtein, V. I., et al. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady* (1966), vol. 10, Soviet Union, pp. 707–710.

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

[91] Li, H. Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. *arXiv preprint arXiv:1303.3997* (2013), 1–3.

[92] Li, H. https://github.com/lh3/bioawk. https://github.com/lh3/bioawk, 2017.

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

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

[95] Li, H., and Durbin, R. Fast and Accurate Short-Read Alignment with Burrows-Wheeler Transform. *Bioinformatics 25*, 14 (2009), 1754–1760.

[96] Li, H., and Durbin, R. Fast and accurate long-read alignment with Burrows–Wheeler transform. *Bioinformatics 26*, 5 (2010), 589–595.

[97] Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., Marth, G., Abecasis, G., and Durbin, R. The sequence alignment/map format and SAMtools. *Bioinformatics 25*, 16 (2009), 2078–2079.

[98] Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., Marth, G., Abecasis, G., Durbin, R., and Subgroup, . G. P. D. P. The sequence alignment/map format and SAMtools. *Bioinformatics 25*, 16 (2009), 2078–2079.

[99] Li, H., and Homer, N. A survey of sequence alignment algorithms for next-generation sequencing. *Briefings in Bioinformatics 11*, 5 (2010), 473–483.

[100] Li, H., Ruan, J., and Durbin, R. Mapping short DNA sequencing reads and calling variants using mapping quality scores. *Genome Research 18*, 11 (2008), 1851–1858.

[101] Li, N., Wang, X., and Liu, Z. Next-Generation Sequencing Technologies and the Assembly of Short Reads into Reference Genome Sequences. *Bioinformatics in Aquaculture: Principles and Methods* (2017).

[102] Li, R., Li, Y., Zheng, H., Luo, R., Zhu, H., Li, Q., Qian, W., Ren, Y., Tian, G., Li, J., et al. Building the sequence map of the human pan-genome. *Nature Biotechnology 28*, 1 (2010), 57–63.

[103] Li, R., Yu, C., Li, Y., Lam, T., Yiu, S., Kristiansen, K., and Wang, J. SOAP2: an improved ultrafast tool for short read alignment. *Bioinformatics 25 15* (2009), 1966–7.

[104] Li, Heng. Burrows-Wheeler Aligner. `https://sourceforge.net/p/bio-bwa/mailman/message/31679354/`, 2013. Accessed: 2018-09.

[105] Liti, G., Carter, D. M., Moses, A. M., Warringer, J., Parts, L., James, S. A., Davey, R. P., Roberts, I. N., Burt, A., Koufopanou, V., et al. Population genomics of domestic and wild yeasts. *Nature 458*, 7236 (2009), 337–341.

[106] Liu, L., Li, Y., Li, S., Hu, N., He, Y., Pong, R., Lin, D., Lu, L., and Law, M. Comparison of next-generation sequencing systems. *Journal of Biomedicine and Biotechnology 2012* (2012).

[107] Lu, H., Giordano, F., and Ning, Z. Oxford Nanopore MinION sequencing and genome assembly. *Genomics, Proteomics & Bioinformatics 14*, 5 (2016), 265–279.

[108] Luikart, G., England, P., Tallmon, D., Jordan, S., and Taberlet, P. The power and promise of population genomics: from genotyping to genome typing. *Nature Reviews Genetics 4*, 12 (2003), 981–994.

[109] Lunter, G., and Goodson, M. Stampy: a statistical algorithm for sensitive and fast mapping of Illumina sequence reads. *Genome Research 21*, 6 (2011), 936–939.

[110] Marçais, G., and Kingsford, C. A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. *Bioinformatics 27*, 6 (2011), 764–770.

[111] Marth, G., Korf, I., Yandell, M., Yeh, R., Gu, Z., Zakeri, H., Stitziel, N., Hillier, L., Kwok, P., and Gish, W. A general approach to single-nucleotide polymorphism discovery. *Nature Genetics 23* (1999), 452–456.

[112] Martin, J. A., and Wang, Z. Next-generation transcriptome assembly. *Nature Reviews Genetics 12*, 10 (2011), 671–682.

[113] Maxam, A. M., and Gilbert, W. A new method for sequencing DNA. *Proceedings of the National Academy of Sciences 74*, 2 (1977), 560–564.

[114] McCarthy, C. G., and Fitzpatrick, D. A. Pan-genome analyses of model fungal species. *Microbial Genomics 5*, 2 (2019).

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

[116] McManus, B. A., and Coleman, D. C. Molecular epidemiology, phylogeny and evolution of *Candida albicans*. *Infection, Genetics and Evolution 21* (2014), 166–178.

[117] McTavish, E. J., Pettengill, J., Davis, S., Rand, H., Strain, E., Allard, M., and Timme, R. E. TreeToReads-a pipeline for simulating raw reads from phylogenies. *BMC Bioinformatics 18*, 1 (2017).

[118] Medini, D., Donati, C., Tettelin, H., Masignani, V., and Rappuoli, R. The microbial pan-genome. *Current Opinion in Genetics & Development 15*, 6 (2005), 589–594.

[119] Mohamadi, H., Chu, J., Vandervalk, B. P., and Birol, I. ntHash: recursive nucleotide hashing. *Bioinformatics 32*, 22 (2016), 3492–3494.

[120] Mu, J. C., Mohiyuddin, M., Li, J., Bani Asadi, N., Gerstein, M. B., Abyzov, A., Wong, W. H., and Lam, H. Y. VarSim: a high-fidelity simulation and validation framework for high-throughput genome sequencing with cancer applications. *Bioinformatics 31*, 9 (2015), 1469–1471.

[121] Needleman, S. B., and Wunsch, C. D. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology 48*, 3 (1970), 443–453.

[122] Neiman, A. M. Ascospore formation in the yeast *Saccharomyces cerevisiae*. *Microbiology and Molecular Biology Reviews 69*, 4 (2005), 565–584.

[123] Nielsen, R., Korneliussen, T., Albrechtsen, A., Li, Y., and Wang, J. SNP calling, genotype calling, and sample allele frequency estimation from new-generation sequencing data. *PLoS ONE 7*, 7 (2012).

[124] Nielsen, R., Paul, J. S., Albrechtsen, A., and Song, Y. S. Genotype and SNP calling from next-generation sequencing data. *Nature Reviews Genetics 12*, 6 (2011), 443–451.

[125] Novak, A. M., Hickey, G., Garrison, E., Blum, S., Connelly, A., Dilthey, A., Eizenga, J., Elmohamed, M. A. S., Guthrie, S., Kahles, A., Keenan, S., Kelleher, J., Kural, D., Li, H., Lin, M. F., Miga, K., Ouyang, N., Rakocevic, G., Smuga-Otto, M., Zaranek, A. W., Durbin, R., McVean, G., Haussler, D., and Paten, B. Genome Graphs. *bioRxiv* (2017).

[126] Nurk, S., Koren, S., Rhie, A., Rautiainen, M., Bzikadze, A. V., Mikheenko, A., Vollger, M. R., Altemose, N., Uralsky, L., Gershman, A., et al. The complete sequence of a human genome. *BioRxiv* (2021).

[127] Olson, N. D., Lund, S. P., Colman, R. E., Foster, J. T., Sahl, J. W., Schupp, J. M., Keim, P., Morrow, J. B., Salit, M. L., Zook, J. M., and et al. Best practices for evaluating single nucleotide variant calling methods for microbial genomics. *Frontiers in Genetics 6* (2015).

[128] Paten, B., Novak, A. M., Eizenga, J. M., and Garrison, E. Genome graphs and the evolution of genome inference. *Genome Research* (2017).

[129] Pattnaik, S., Gupta, S., Rao, A. A., and Panda, B. SInC: an accurate and fast error-model based simulator for SNPs, Indels and CNVs coupled with a read generator for short-read sequence data. *BMC Bioinformatics 15*, 1 (2014), 1–9.

[130] Peter, J., De Chiara, M., Friedrich, A., Yue, J.-X., Pflieger, D., Bergström, A., Sigwalt, A., Barre, B., Freel, K., Llored, A., et al. Genome evolution across 1,011 Saccharomyces cerevisiae isolates. *Nature 556*, 7701 (2018), 339–344.

[131] Pirooznia, M., Kramer, M., Parla, J., Goes, F. S., Potash, J. B., Mccombie, W., and Zandi, P. P. Validation and assessment of variant calling pipelines for next-generation sequencing. *Human Genomics 8*, 1 (2014), 14.

[132] Piškur, J., and Langkjaer, R. B. Yeast genome sequencing: the power of comparative genomics. *Molecular Microbiology 53*, 2 (2004), 381–389.

[133] Plissonneau, C., Hartmann, F. E., and Croll, D. Pangenome analyses of the wheat pathogen *Zymoseptoria tritici* reveal the structural basis of a highly plastic eukaryotic genome. *BMC Biology 16*, 1 (2018), 1–16.

[134] Poplin, R., Ruano-Rubio, V., Depristo, M. A., Fennell, T. J., Carneiro, M. O., Auwera, G. A. V. D., Kling, D. E., Gauthier, L. D., Levy-Moonshine, A., Roazen, D., and et al. Scaling accurate genetic variant discovery to tens of thousands of samples. *bioRxiv* (2017).

[135] Priestley, P., Baber, J., Lolkema, M. P., Steeghs, N., de Bruijn, E., Shale, C., Duyvesteyn, K., Haidari, S., van Hoeck, A., Onstenk, W., et al. Pan-cancer whole-genome analyses of metastatic solid tumours. *Nature 575*, 7781 (2019), 210–216.

[136] Quick, J., Loman, N. J., Duraffour, S., Simpson, J. T., Severi, E., Cowley, L., Bore, J. A., Koundouno, R., Dudas, G., Mikhail, A., and et al. Real-time, portable genome sequencing for Ebola surveillance. *Nature 530*, 7589 (2016), 228–232.

[137] R Core Team. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, 2018.

[138] Rakocevic, G., Semenyuk, V., Lee, W.-P., Spencer, J., Browning, J., Johnson, I. J., Arsenijevic, V., Nadj, J., Ghose, K., Suciu, M. C., Ji,

S.-G., Demir, G., Li, L., Toptaş, B. C., Dolgoborodov, A., Pollex, B., Spulber, I., Glotova, I., Kómár, P., Stachyra, A. L., Li, Y., Popovic, M., Källberg, M., Jain, A., and Kural, D. Fast and accurate genomic analyses using genome graphs. *Nature Genetics* (Jan 2019).

[139] Raman, R., Raman, V., and Satti, S. R. Succinct indexable dictionaries with applications to encoding k-ary trees, prefix sums and multisets. *ACM Transactions on Algorithms 3*, 4 (2007).

[140] Rambaut, A., and Grass, N. C. Seq-Gen: an application for the Monte Carlo simulation of DNA sequence evolution along phylogenetic trees. *Bioinformatics 13*, 3 (06 1997), 235–238.

[141] Rimmer, A., Phan, H., Mathieson, I., Iqbal, Z., Twigg, S. R. F., Wilkie, A. O. M., Mcvean, G., and Lunter, G. Integrating mapping-, assembly-and haplotype-based approaches for calling variants in clinical sequencing applications. *Nature Genetics 46*, 8 (2014), 912–918.

[142] Saccone, C., and Pesole, G. *Handbook of comparative genomics: Principles and Methodology*. John Wiley & Sons, 2003.

[143] Sandmann, S., Graaf, A. O. D., Karimi, M., Reijden, B. A. V. D., Hellström-Lindberg, E., Jansen, J. H., and Dugas, M. Evaluating variant calling tools for non-matched next-generation sequencing data. *Scientific Reports 7*, 1 (2017).

[144] Sanger, F., Air, G., Barrell, B., Brown, N., Coulson, A., Fiddes, C., Hutchison, C., Slocombe, P., and Smith, M. Nucleotide sequence of bacteriophage φX174 DNA. *Nature 265*, 5596 (1977), 687–695.

[145] Sanger, F., Nicklen, S., and Coulson, A. R. DNA sequencing with chain-terminating inhibitors. *Proceedings of the National Academy of Sciences 74*, 12 (1977), 5463–5467.

[146] Saraswathy, N., and Ramalingam, P. Genomes of Model Organisms. In *Concepts and Techniques in Genomics and Proteomics*, N. Saraswathy and P. Ramalingam, Eds., Woodhead Publishing Series in Biomedicine. Woodhead Publishing, 2011, pp. 29 – 48.

[147] Schadt, E. E., Turner, S., and Kasarskis, A. A window into third-generation sequencing. *Human Molecular Genetics 19*, 2 (2010), 227–R240.

[148] Schbath, S., Martin, V., Zytnicki, M., Fayolle, J., Loux, V., and Gibrat, J.-F. Mapping reads on a genomic sequence: an algorithmic overview and a practical comparative analysis. *Journal of Computational Biology 19*, 6 (2012), 796–813.

[149] Schneider, V. A., Lindsay, T. G., Howe, K., Bouk, N., Chen, H.-C., Kitts, P. A., Murphy, T. D., Pruitt, K. D., Thibaud-Nissen, F., Albracht, D., and et al. Evaluation of GRCh38 and de novo haploid genome assemblies demonstrates the enduring quality of the reference assembly. *Genome Research 27*, 5 (2017), 849–864.

[150] Sibbesen, J. A., Maretty, L., Krogh, A., and Consortium, T. D. P.-G. Accurate genotyping across variant classes and lengths using variant graphs. *Nature Genetics 50*, 7 (2018), 1054–1059.

[151] Smith, T., and Waterman, M. Identification of common molecular subsequences. *Journal of Molecular Biology 147*, 1 (1981), 195–197.

[152] Solieri, L. Mitochondrial inheritance in budding yeasts: towards an integrated understanding. *Trends in Microbiology 18*, 11 (2010), 521–530.

[153] Stappers, M., and Brown, G. *Candida albicans: Cellular and Molecular Biology*. Springer, Cham, 2017.

[154] Stein, A., Takasuka, T. E., and Collings, C. K. Are nucleosome positions in vivo primarily determined by histone–DNA sequence preferences? *Nucleic Acids Research 38*, 3 (2010), 709–719.

[155] Swerdlow, H., and Gesteland, R. Capillary gel electrophoresis for rapid, high resolution DNA sequencing. *Nucleic Acids Research 18*, 6 (1990), 1415–1419.

[156] Swofford, D. *PAUP*: phylogenetic analysis using parsimony (*and Other Methods). Version 4.0b10*, vol. Version 4.0. Sinauer Associates, 01 2002.

[157] Taliun, D., Harris, D. N., Kessler, M. D., Carlson, J., Szpiech, Z. A., Torres, R., Taliun, S. A. G., Corvelo, A., Gogarten, S. M., Kang, H. M., et al. Sequencing of 53,831 diverse genomes from the NHLBI TOPMed Program. *Nature 590*, 7845 (2021), 290–299.

[158] Tan, A., Abecasis, G. R., and Kang, H. M. Unified representation of genetic variants. *Bioinformatics 31*, 13 (2015), 2202–2204.

[159] Tavaré, S., and Miura, R. M. Lectures on mathematics in the life sciences. In *American Mathematical Society* (1986), pp. 57–86.

[160] The 1002 Yeast Genome Project. The 1002 Yeast Genome Project Files. `http://1002genomes.u-strasbg.fr/files/`, 2019. Accessed: 2018-05-01.

[161] Thompson, G., Patel, P., Kirkpatrick, W., Westbrook, S., Berg, D., Erlandsen, J., Redding, S., and Patterson, T. *Oropharyngeal candidiasis* in the era of antiretroviral therapy. *Oral Surg Oral Med Oral Pathol Oral Radiol Endod 109*, 4 (2010), 488–95.

[162] Van der Auwera, G. A., Carneiro, M. O., Hartl, C., Poplin, R., del Angel, G., Levy-Moonshine, A., Jordan, T., Shakir, K., Roazen, D., Thibault, J., Banks, E., Garimella, K. V., Altshuler, D., Gabriel, S., and DePristo, M. A. From FastQ data to high-confidence variant calls: the genome analysis toolkit best practices pipeline. *Current Protocols in Bioinformatics 43*, 1 (2013), 11.10.1–11.10.33.

[163] Vernikos, G., Medini, D., Riley, D. R., and Tettelin, H. en years of pan-genome analyses. *Current Opinion in Microbiology 23* (2015), 148–154.

[164] Wang, Q., Pierce-Hoffman, E., Cummings, B. B., Alföldi, J., Francioli, L. C., Gauthier, L. D., Hill, A. J., O'Donnell-Luria, A. H., Karczewski, K. J., and MacArthur, D. G. Landscape of multi-nucleotide variants in 125,748 human exomes and 15,708 genomes. *Nature Communications 11*, 1 (2020), 1–13.

[165] Weischenfeldt, J., Symmons, O., Spitz, F., and Korbel, J. O. Phenotypic impact of genomic structural variation: insights from and for human disease. *Nature Reviews Genetics 14*, 2 (2013), 125–138.

[166] Willems, T., Zielinski, D., Yuan, J., Gordon, A., Gymrek, M., and Erlich, Y. Genome-wide profiling of heritable and de novo STR variations. *Nature Methods 14*, 6 (2017), 590–592.

[167] Williamson, D. The curious history of yeast mitochondrial DNA. *Nature Reviews Genetics 3*, 6 (2002), 475–481.

[168] Wolters, J. F., Chiu, K., and Fiumera, H. L. Population structure of mitochondrial genomes in *Saccharomyces cerevisiae*. *BMC Genomics 16*, 1 (2015), 1–13.

[169] Xin, H., Lee, D., Hormozdiari, F., Yedkar, S., Mutlu, O., and Alkan, C. Accelerating read mapping with FastHASH. *BMC Genomics 14*, 1 (2013), 1–13.

[170] Yao, W., Li, G., Zhao, H., Wang, G., Lian, X., and Xie, W. Exploring the rice dispensable genome using a metagenome-like assembly strategy. *Genome Biology 16*, 1 (2015), 1–20.

[171] Zhang, C., and Ochoa, I. VEF: a variant filtering tool based on ensemble methods. *Bioinformatics 36*, 8 (2019), 2328–2336.

[172] Zhu, Y. O., Sherlock, G., and Petrov, D. A. Whole genome analysis of 132 clinical Saccharomyces cerevisiae strains reveals extensive ploidy variation. *G3: Genes, Genomes, Genetics 6*, 8 (2016), 2421–2434.

[173] Zinjarde, S., Apte, M., Mohite, P., and Kumar, A. R. *Yarrowia lipolytica* and pollutants: interactions and applications. *Biotechnology Advances 32*, 5 (2014), 920 – 933.

[174] Zook, J. M., Chapman, B., Wang, J., Mittelman, D., Hofmann, O., Hide, W., and Salit, M. Integrating human sequence data sets provides a resource of benchmark SNP and indel genotype calls. *Nat Biotechnology 32*, 3 (2014), 246–51.

Appendix A

Variation Graphs

Table A.1: **Variation Graph Statistics.** The number of nodes and the length of sequences before (BP) and after (AP) pruning for the NCYC variation graph containing 9 *S. cerevisiae* strains and the Bergstrom variation graph containing 20 strains.

Chromosome	**Graph**	**NCYC Graph**		**Bergstrom Graph**	
		BP	**AP**	**BP**	**AP**
S288c_ChrI	nodes	16727	13960	32474	28108
	length	240975	235172	240562	236016
S288c_ChrII	nodes	31029	29503	74257	70159
	length	827500	824261	833502	829104
S288c_ChrIII	nodes	9995	9250	34505	31442
	length	322216	320148	326992	323655
S288c_ChrIV	nodes	52998	49971	144987	135176
	length	1556955	1550526	1572375	1561952
S288c_ChrV	nodes	24575	23136	53945	50291
	length	588596	585524	591935	588088
S288c_ChrVI	nodes	13972	12916	31145	27913
	length	277529	275164	279715	276321
S288c_ChrVII	nodes	38714	36454	108485	99904
	length	1110131	1105106	1122141	1112992
S288c_ChrVIII	nodes	20442	18861	57968	52896
	length	573151	569927	579474	574126
S288c_ChrIX	nodes	31665	27765	52755	45985
	length	456911	449403	456146	449010
S288c_ChrX	nodes	29647	27816	76737	69687
	length	759405	755692	768077	760684
S288c_ChrXI	nodes	25003	23901	56943	54723
	length	678228	675793	682352	679799
S288c_ChrXII	nodes	36297	34542	144268	112524
	length	1095062	1091435	1131450	1099079
S288c_ChrXIII	nodes	28374	27009	83373	78547
	length	937546	934570	947264	942139
S288c_ChrXIV	nodes	25041	23526	72987	68294
	length	796423	793144	804607	799602
S288c_ChrXV	nodes	40545	38027	106756	99103
	length	1110627	1105323	1121335	1113270
S288c_ChrXVI	nodes	31406	29627	89027	82862
	length	962487	958970	972699	966167
S288c_Chrmt	nodes	12923	10232	12488	10643
	length	101164	92620	92233	89349

Appendix B

Graph Genome Software Comparison

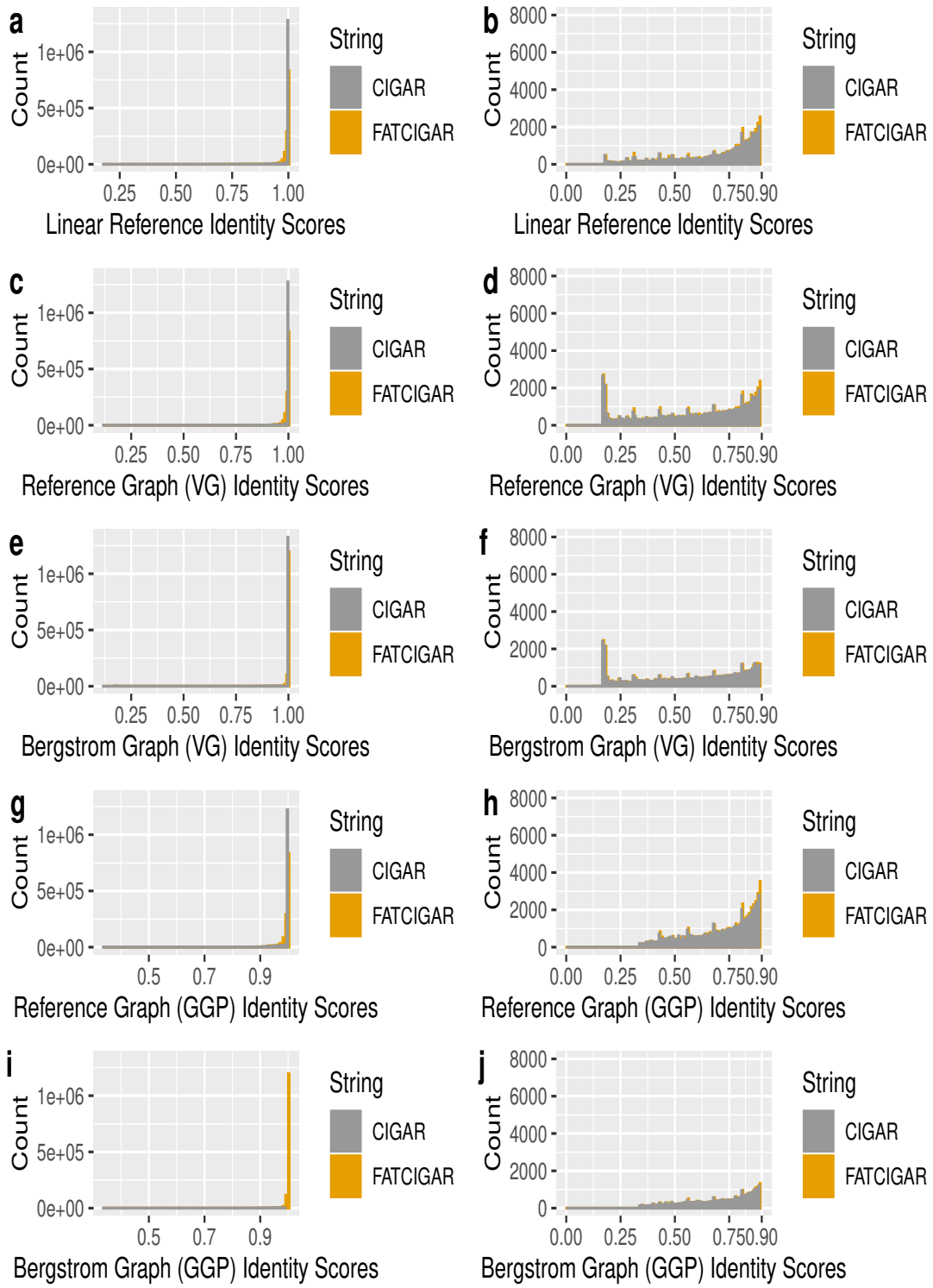


Figure B.1: **Sequence Identity Score Histogram for BC187.**

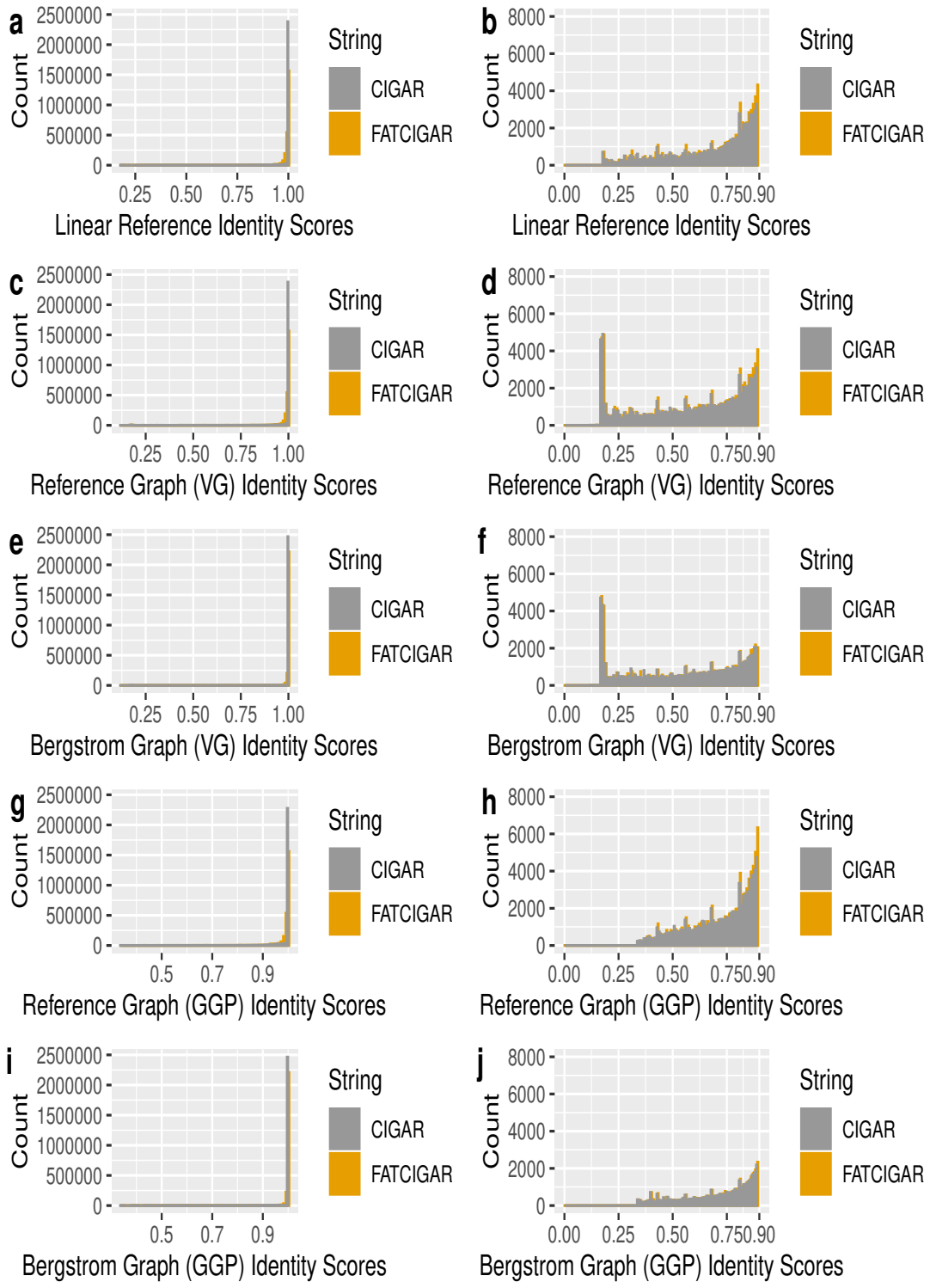


Figure B.2: **Sequence Identity Score Histogram for DBVPG1373.**

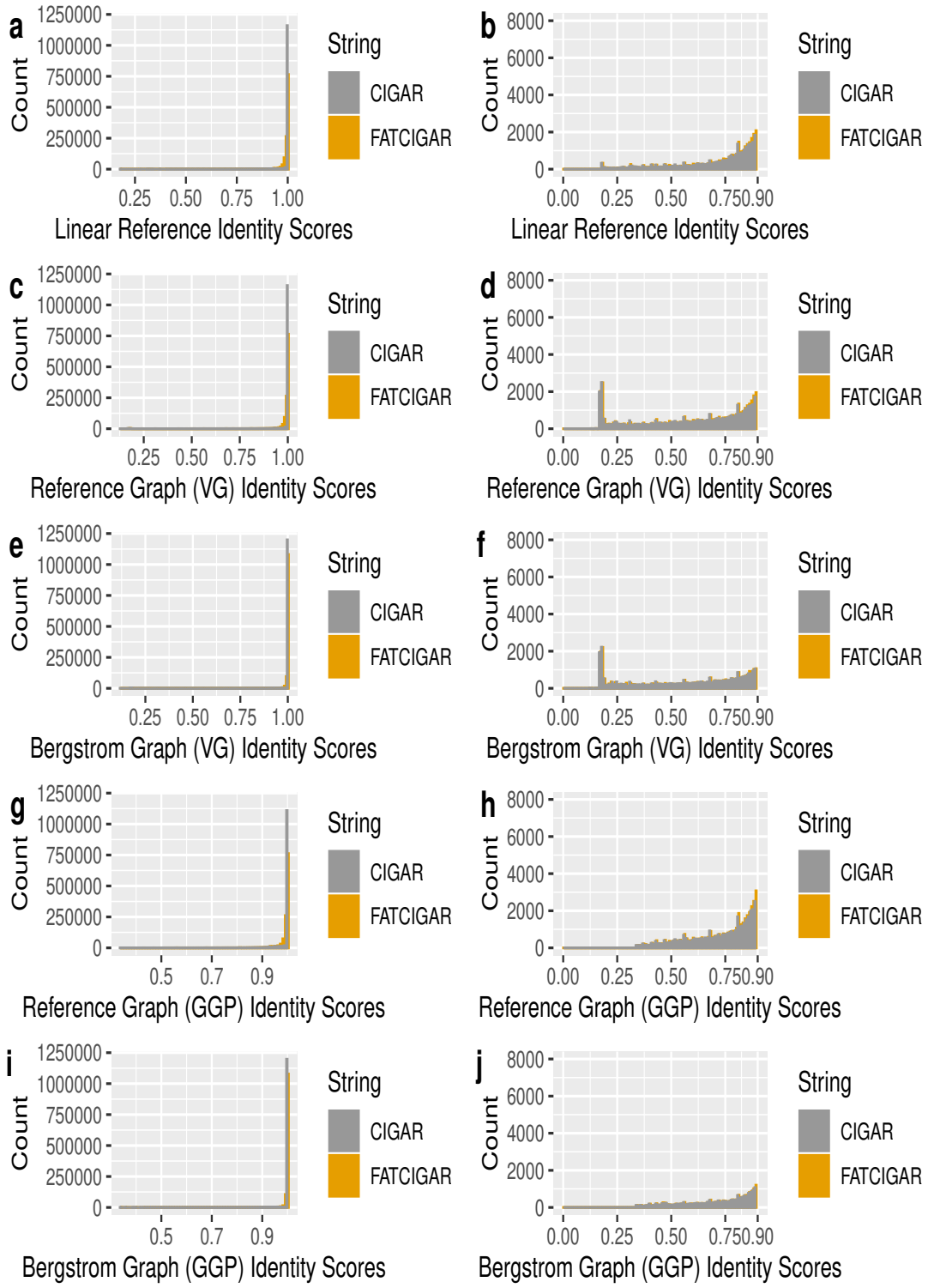


Figure B.3: **Sequence Identity Score Histogram for DBVPG1788.**

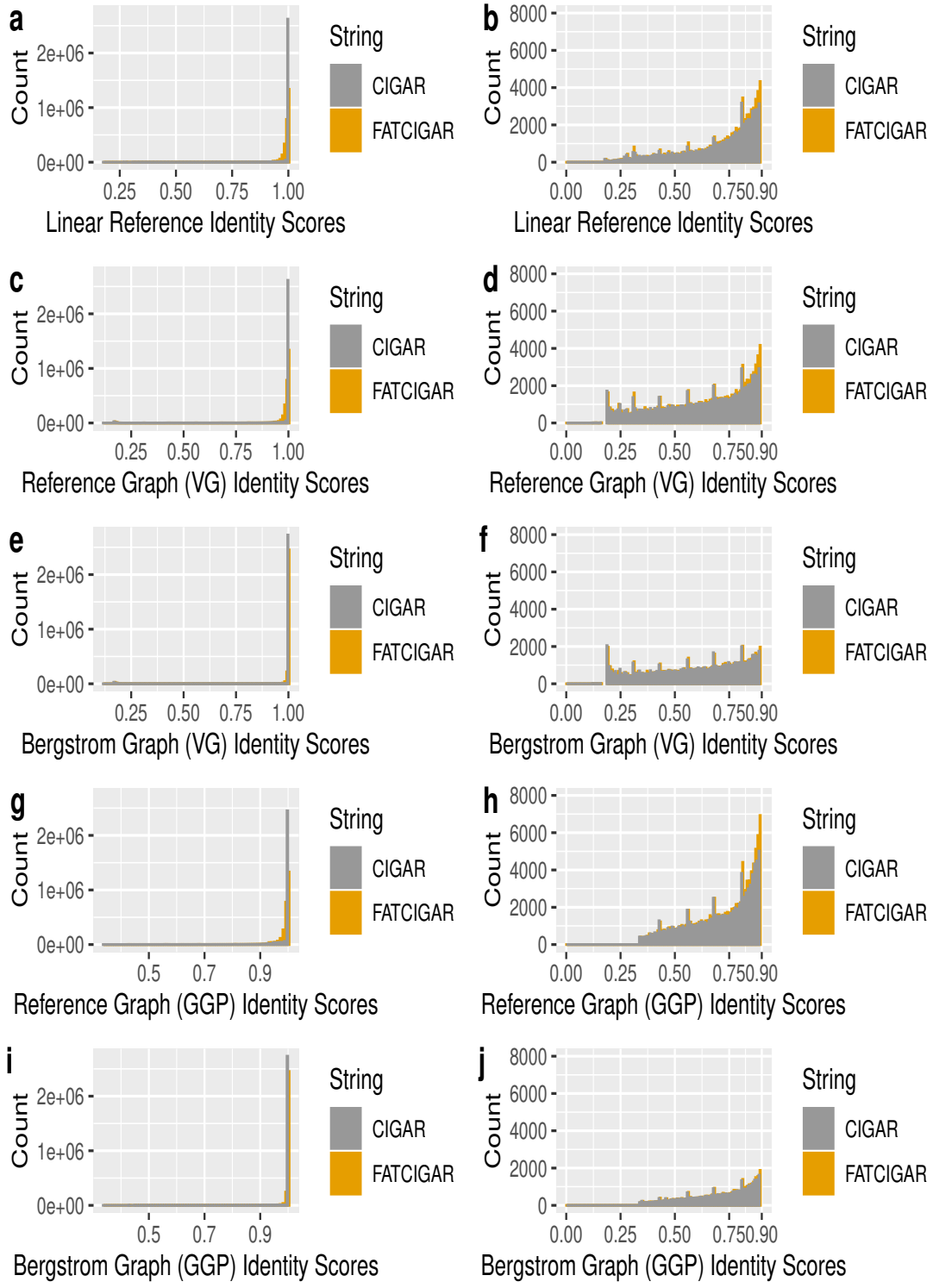


Figure B.4: **Sequence Identity Score Histogram for DBVPG6044.**

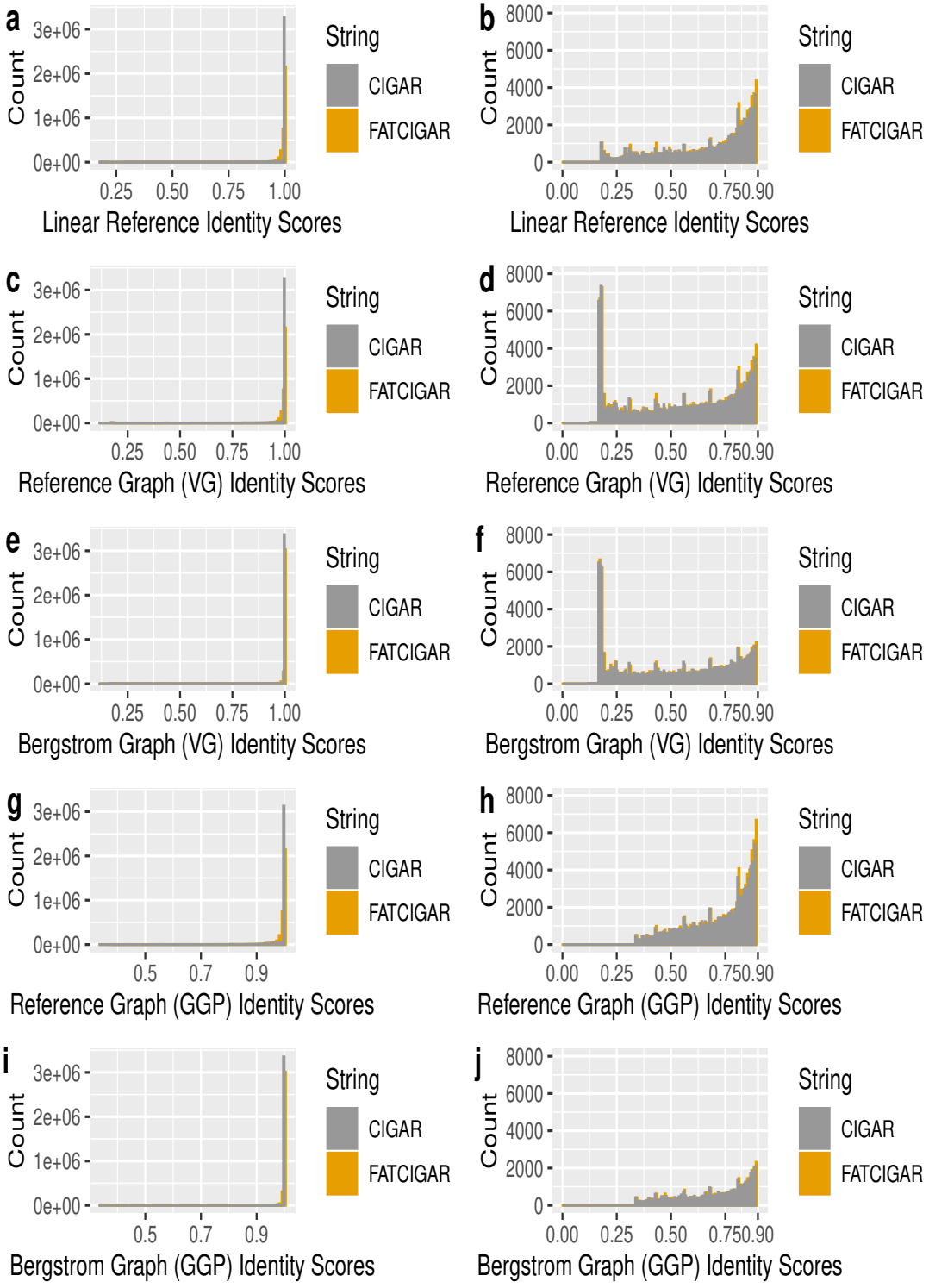


Figure B.5: **Sequence Identity Score Histogram for DBVPG6765.**

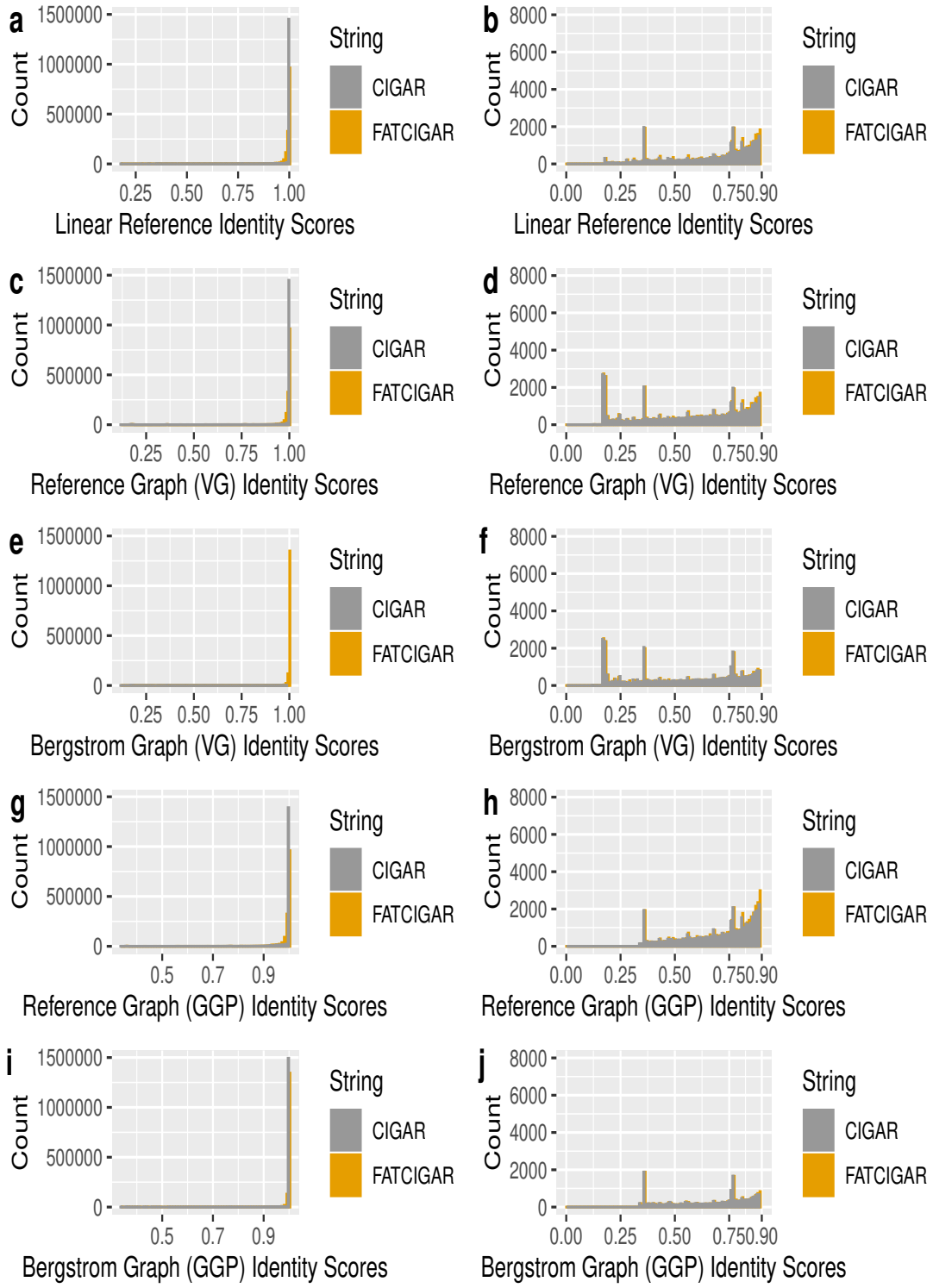


Figure B.6: **Sequence Identity Score Histogram for L1374.**

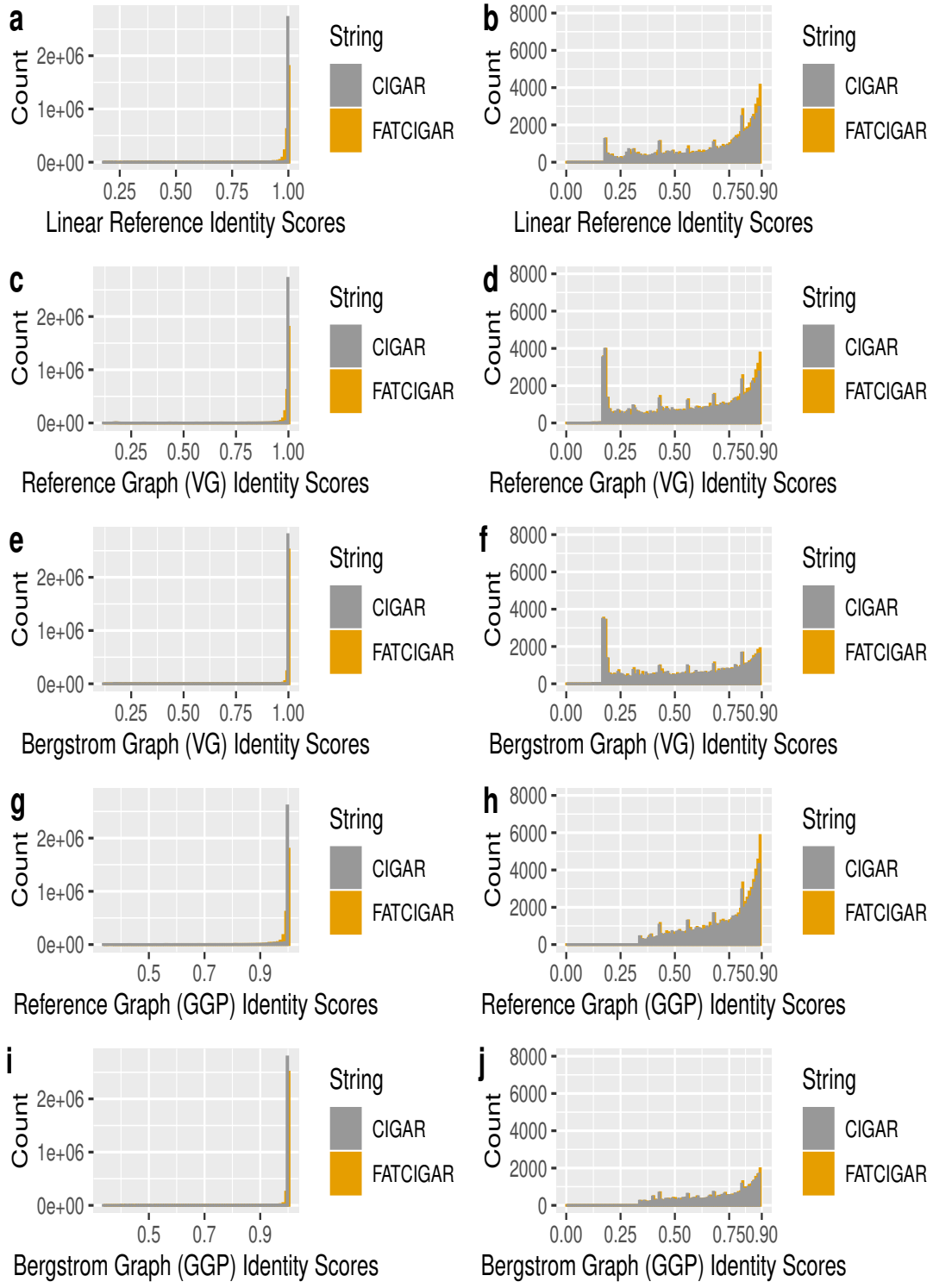


Figure B.7: **Sequence Identity Score Histogram for L1528.**

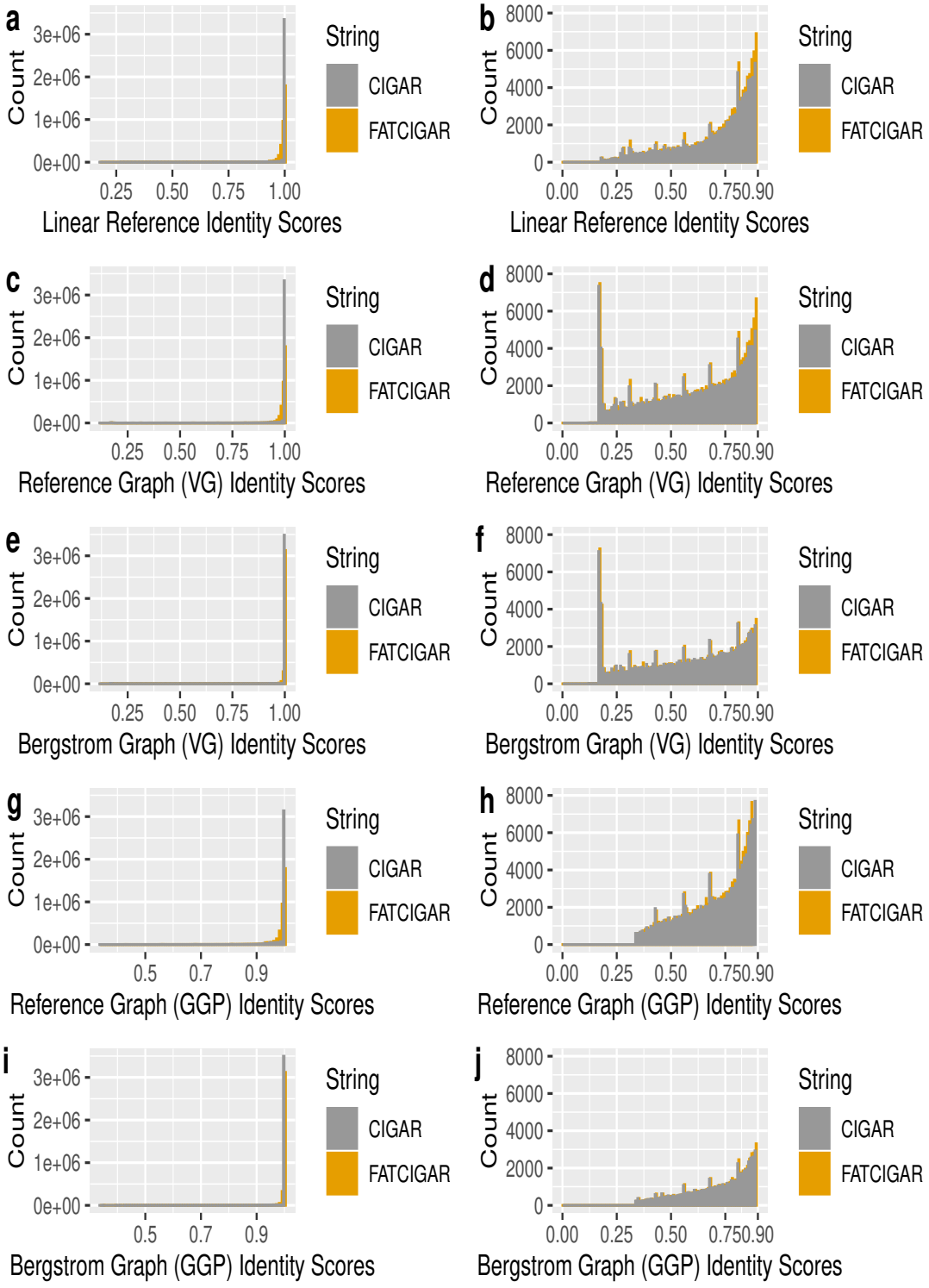


Figure B.8: **Sequence Identity Score Histogram for SK1.**

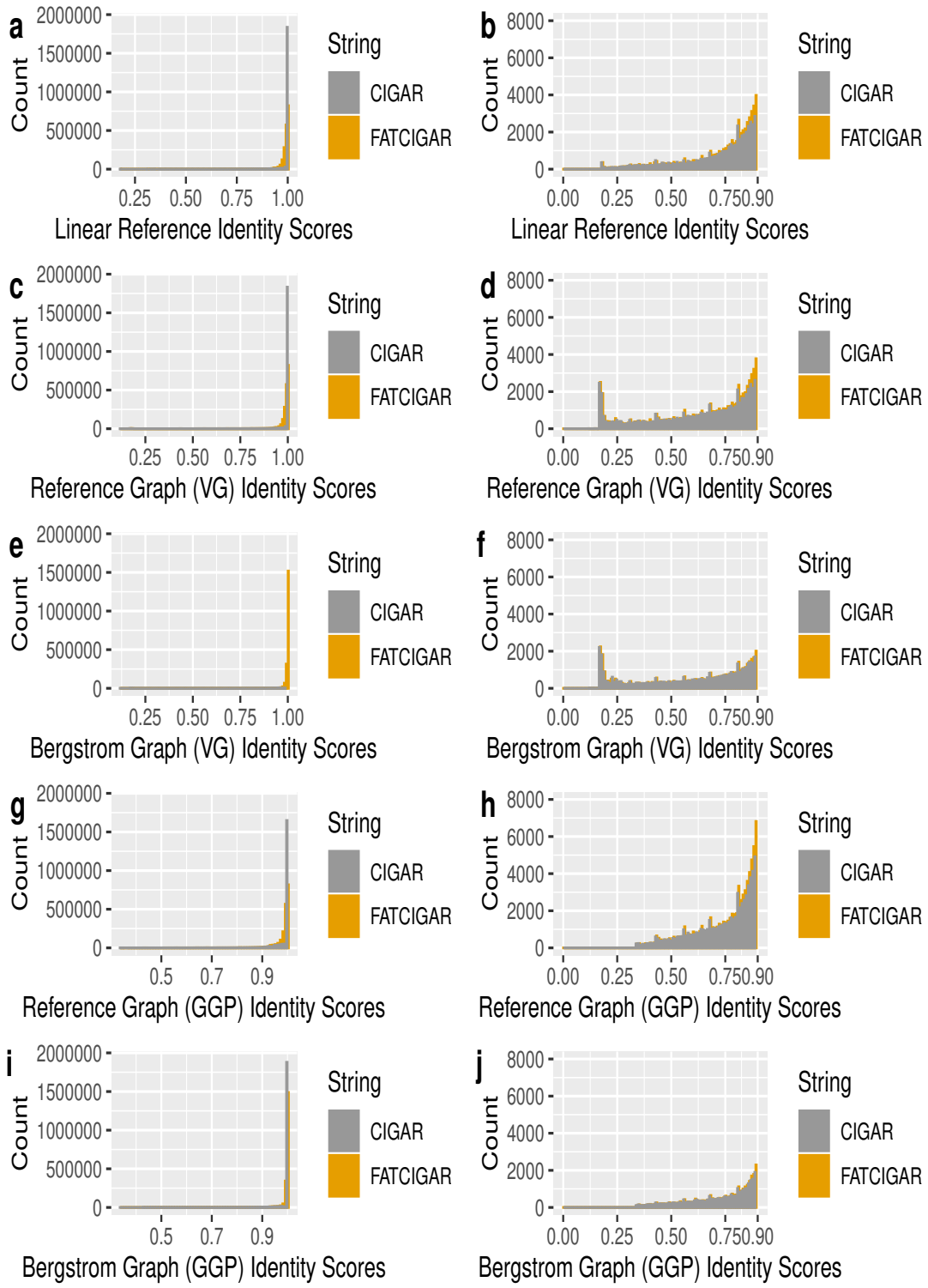


Figure B.9: **Sequence Identity Score Histogram for UWOPS03-461.4.**

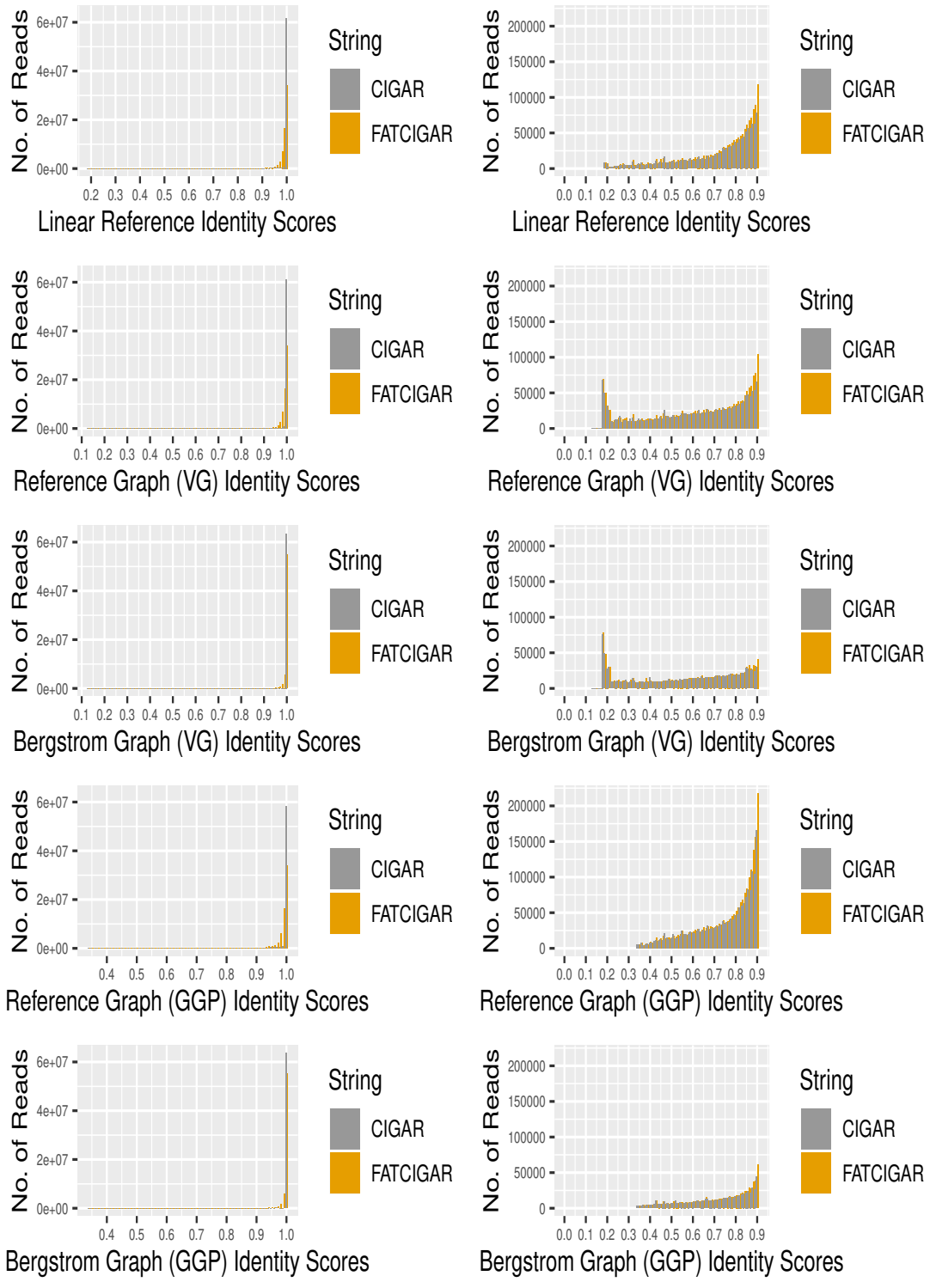


Figure B.10: **Sequence Identity Score Histogram for UWOPS83-787.3.**

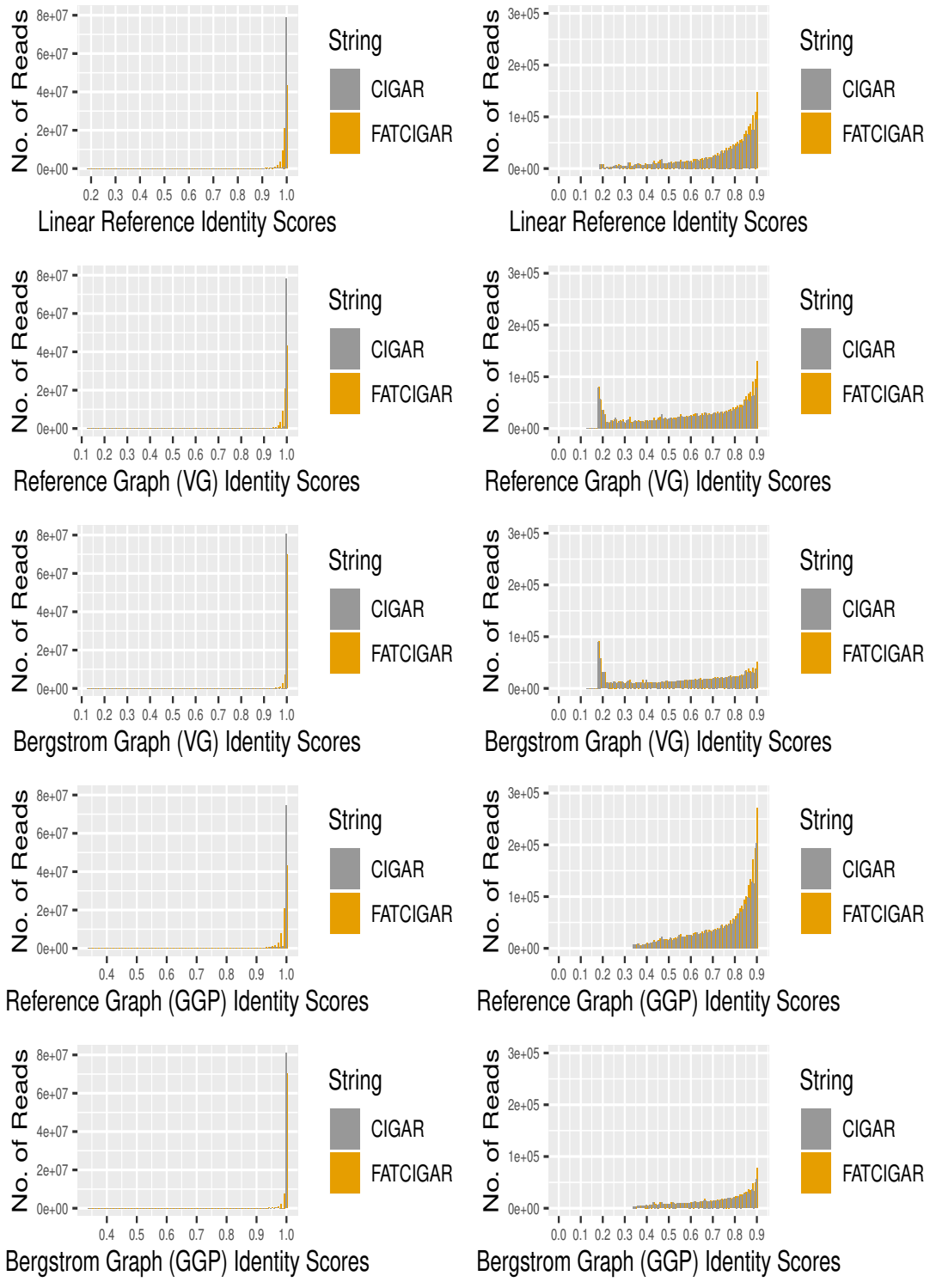


Figure B.11: **Sequence Identity Score Histogram for UWOPS87-2421.**

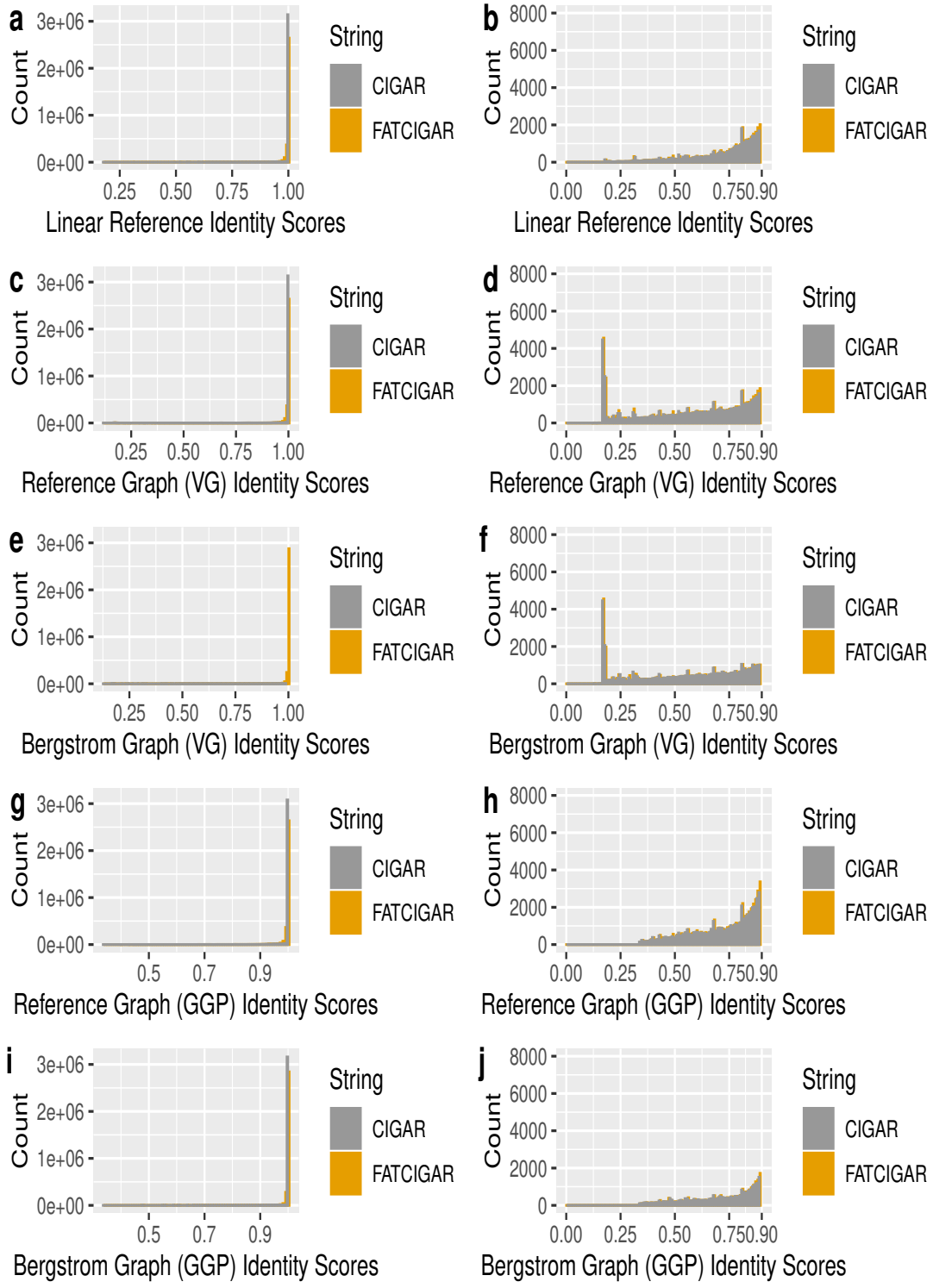


Figure B.12: **Sequence Identity Score Histogram for W303.**

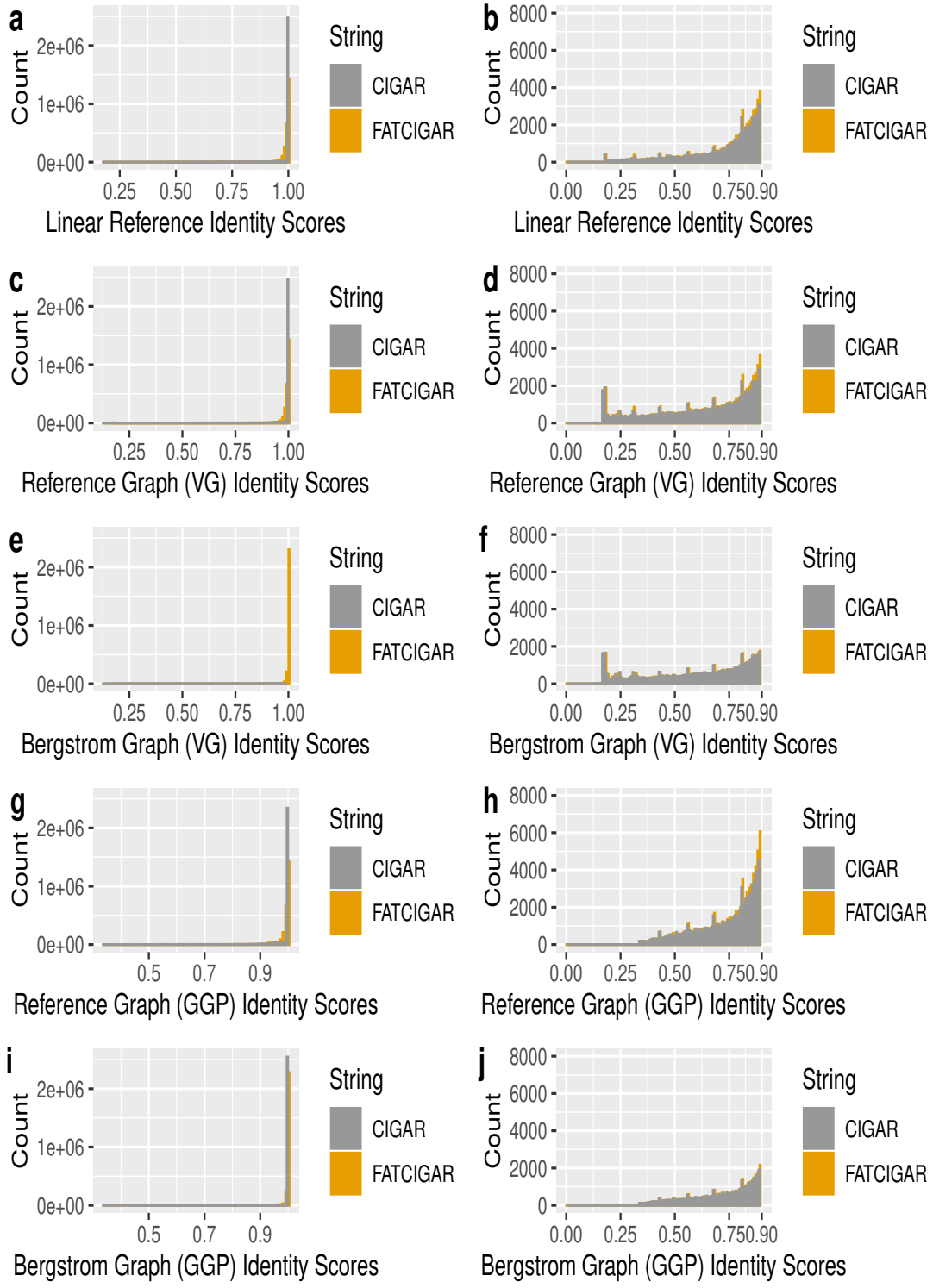


Figure B.13: **Sequence Identity Score Histogram for Y12.**

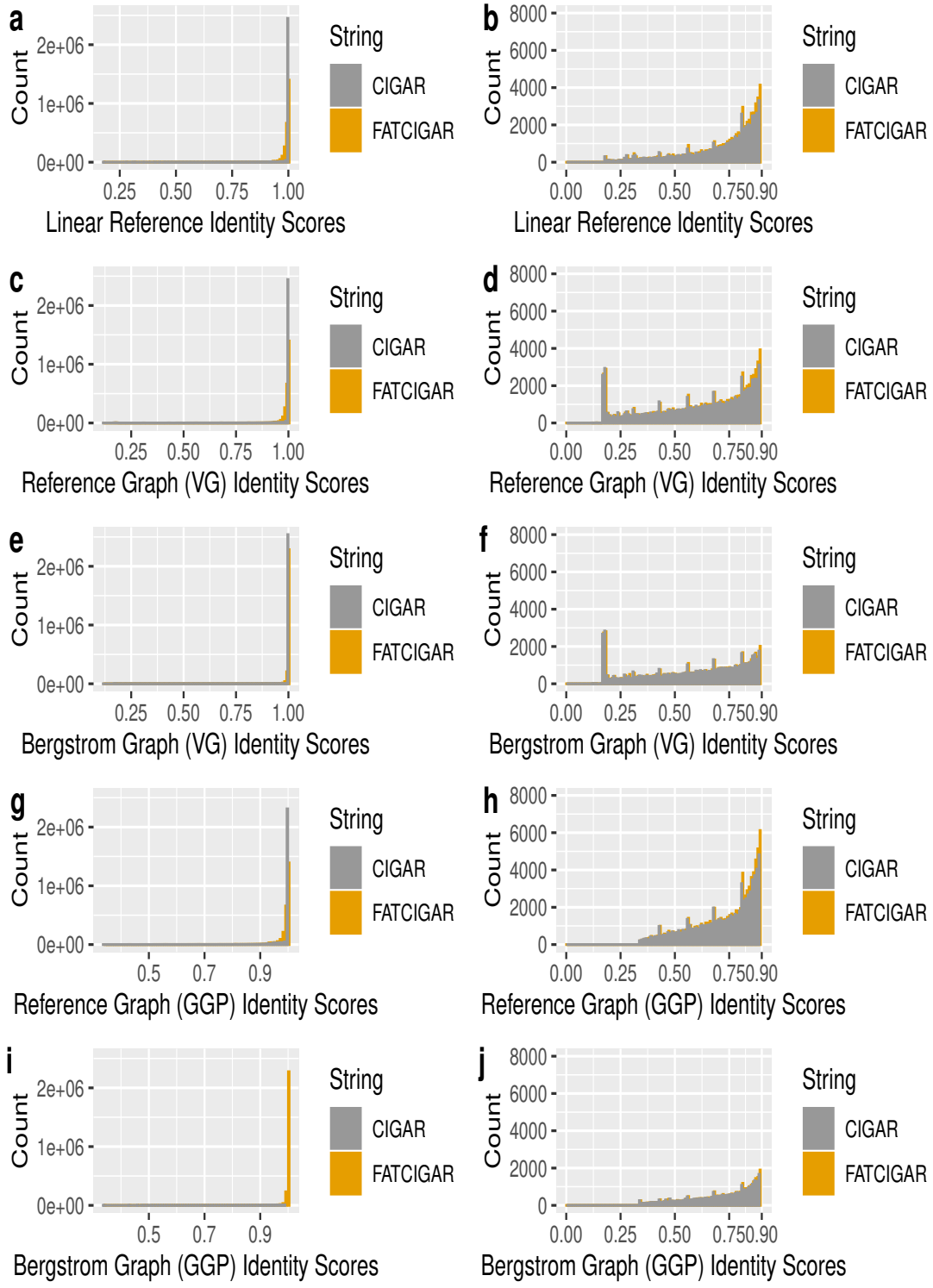


Figure B.14: **Sequence Identity Score Histogram for Y55.**

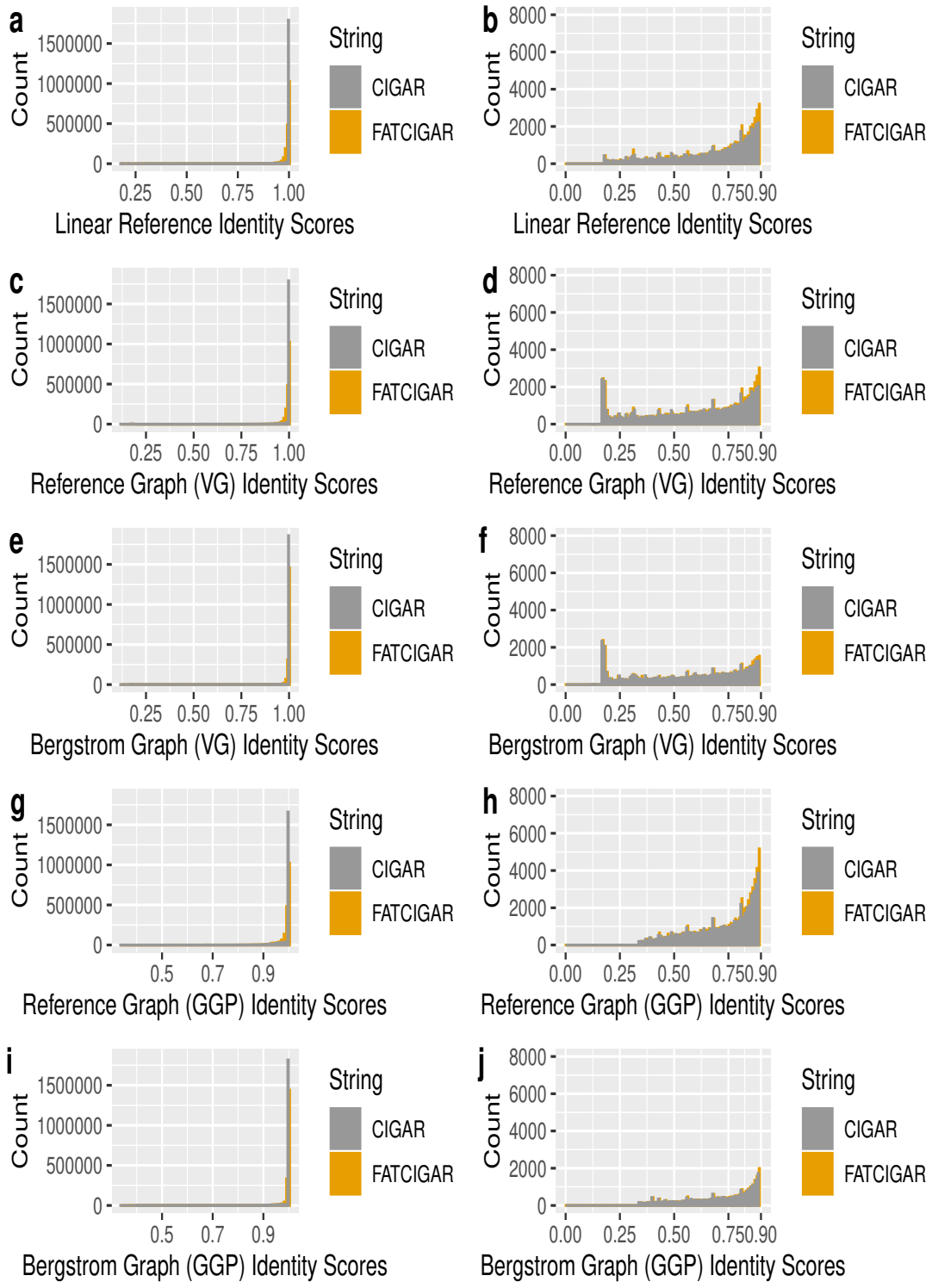


Figure B.15: **Sequence Identity Score Histogram for YJM975.**

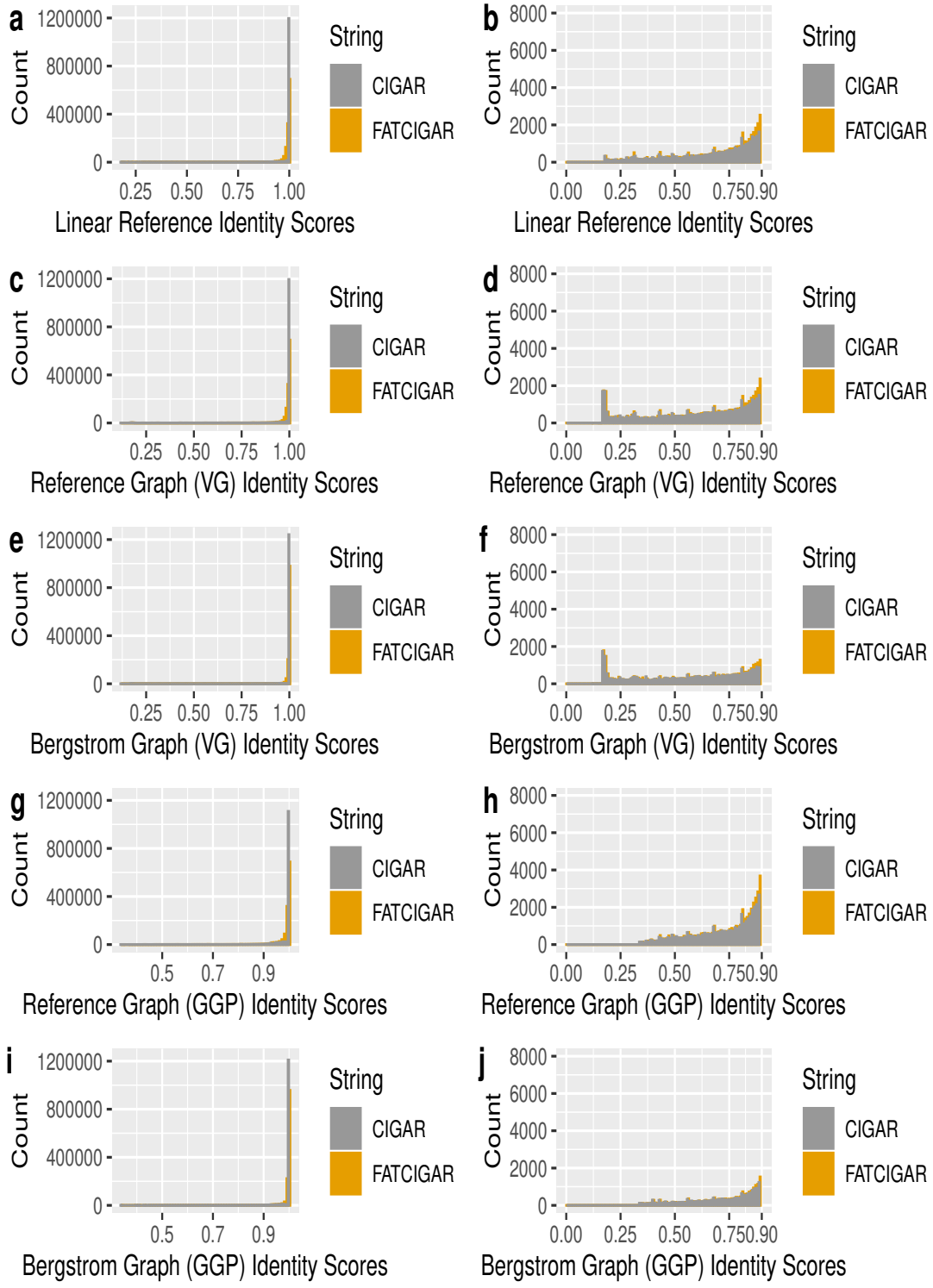


Figure B.16: **Sequence Identity Score Histogram for YJM978.**

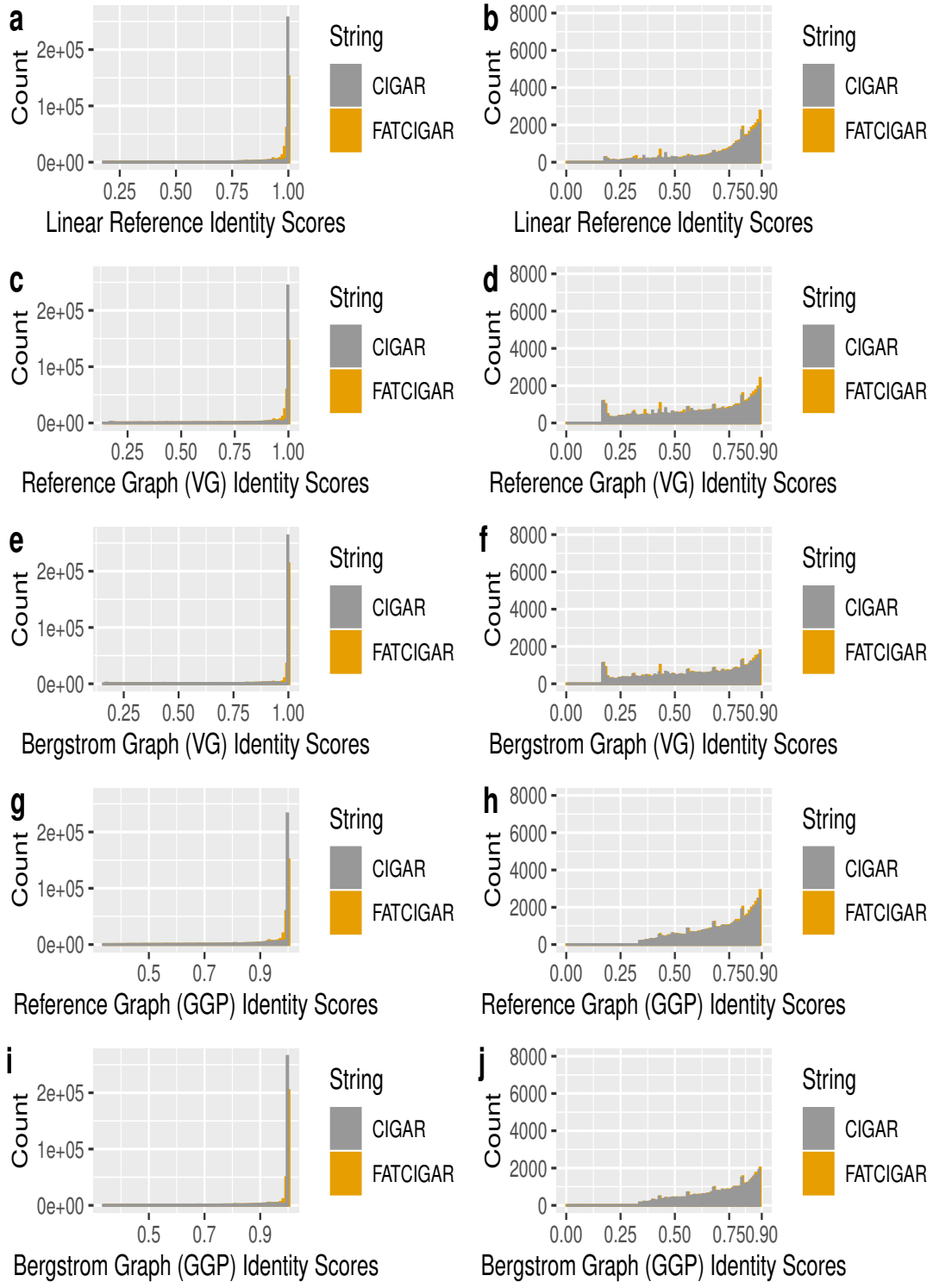


Figure B.17: **Sequence Identity Score Histogram for YJM981.**

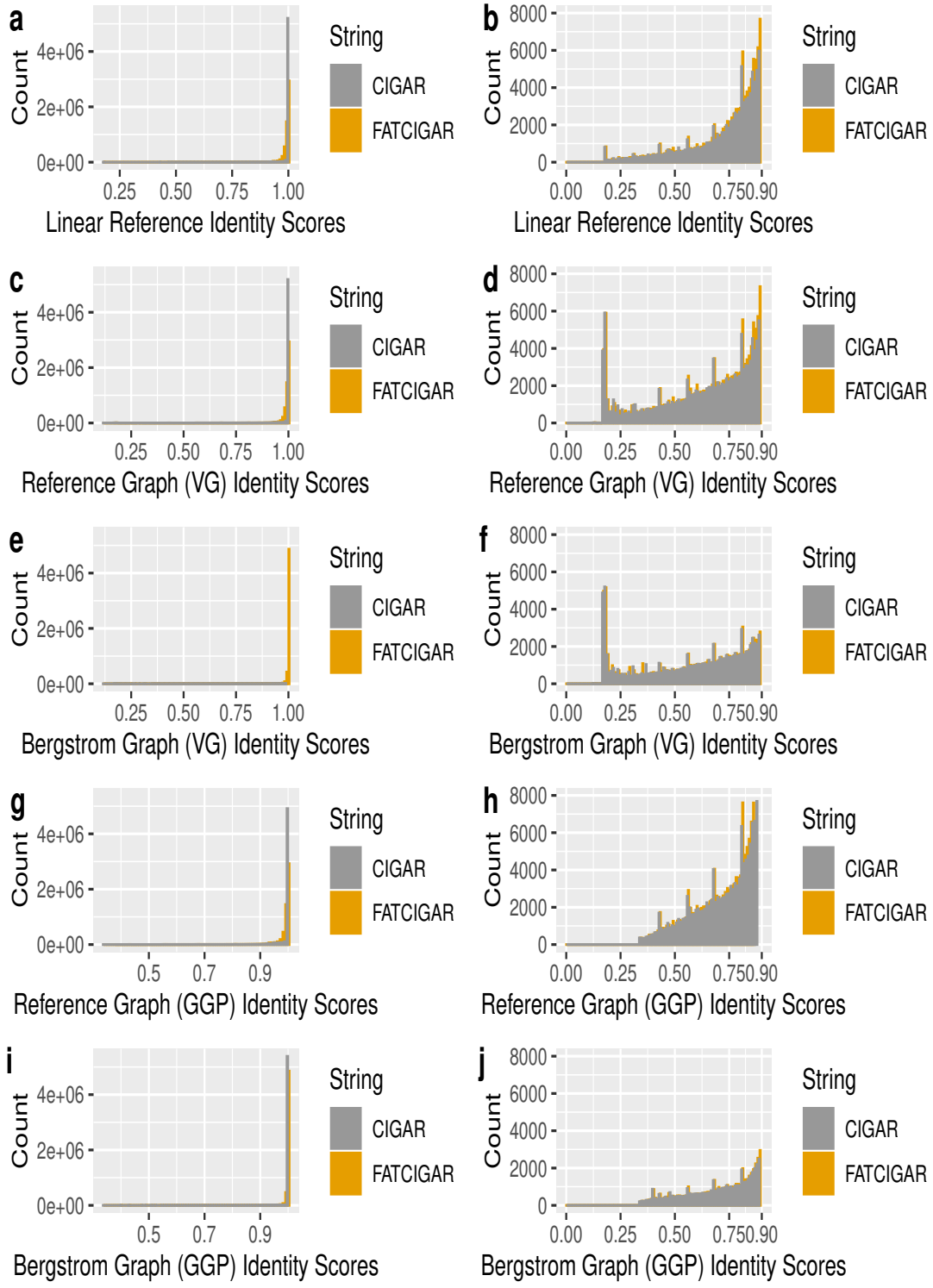


Figure B.18: **Sequence Identity Score Histogram for YPS128.**

Table B.1: **Heuristic Testing of the *Graphtyper* Variant Calling Algorithm.** Number of variants called against the *Graphtyper* reference graph for the simulated Bergstrom strains in Set 1 across 10 repeated runs.

Strains	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9	Run 10
DBVPG1373	4753	4753	4753	4753	4753	4753	4753	4753	4753	4753
DBVPG6044	3099	3099	3099	3099	3099	3099	3099	3099	3099	3099
L1374	2853	2853	2853	2853	2853	2853	2853	2853	2853	2853
SK1	2966	2966	2966	2966	2966	2966	2966	2966	2966	2966
UWOPS83-787.3	2022	2022	2022	2022	2022	2022	2022	2022	2022	2022
W303	413	413	413	413	413	413	413	413	413	413
Y55	2089	2089	2089	2089	2089	2089	2089	2089	2089	2089
YJM978	4002	4002	4002	4002	4002	4002	4002	4002	4002	4002
YPS128	2574	2574	2574	2574	2574	2574	2574	2574	2574	2574
BC187	572	572	572	572	572	572	572	572	572	572
DBVPG1106	4697	4697	4697	4697	4697	4697	4697	4697	4697	4697
DBVPG1788	3647	3647	3647	3647	3647	3647	3647	3647	3647	3647
DBVPG6765	3717	3717	3717	3717	3717	3717	3717	3717	3717	3717
L1528	2966	2966	2966	2966	2966	2966	2966	2966	2966	2966
UWOPS03-461.4	3194	3194	3194	3194	3194	3194	3194	3194	3194	3194
UWOPS87-2421	2286	2286	2286	2286	2286	2286	2286	2286	2286	2286
Y12	2839	2839	2839	2839	2839	2839	2839	2839	2839	2839
YJM975	3955	3955	3955	3955	3955	3955	3955	3955	3955	3955
YJM981	4031	4031	4031	4031	4031	4031	4031	4031	4031	4031

Table B.2: **Heuristic Testing of the *vg* v1.26 Variant Calling Algorithm.** Number of variants called against the *vg* reference graph for the simulated Bergstrom strains in Set 1 across 10 repeated runs. Runs where the number of variants called differ from the majority call have been highlighted in bold.

Strains	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9	Run 10
DBVPG1373	4714	4714	4714	4714	4714	4714	4714	4714	4714	4714
DBVPG6044	3102	3102	3102	3102	3102	3102	3102	3102	3102	3102
L1374	2852	2852	2852	2852	2852	2852	2852	2852	2852	2852
SK1	2961	2961	2961	2961	2961	2961	2961	2961	2961	2961
UWOPS83-787.3	2041	2041	2041	2041	2041	2041	2041	2041	2041	2041
W303	424	424	424	424	424	424	424	424	424	424
Y55	2091	2091	2091	2091	2091	2091	2091	2091	2091	2091
YJM978	3981	3981	3981	3981	3981	3981	3981	3981	3981	3981
YPS128	2584	2584	2584	2584	**2585**	2584	2584	2584	2584	**2585**
BC187	580	580	580	580	580	580	580	580	580	580
DBVPG1106	4646	4646	4646	4646	4646	4646	4646	4646	4646	4646
DBVPG1788	3623	3623	3623	3623	3623	3623	3623	3623	3623	3623
DBVPG6765	3686	3686	3686	3686	3686	3686	3686	3686	3686	3686
L1528	2977	2977	2977	2977	2977	2977	2977	2977	2977	2977
UWOPS03-461.4	3175	3175	3175	3175	3175	3175	3175	3175	3175	3175
UWOPS87-2421	2276	2276	2276	2276	2276	2276	2276	2276	2276	2276
Y12	2841	2841	2841	2841	2841	2841	2841	2841	2841	2841
YJM975	3940	3940	3940	3940	3940	3940	3940	3940	3940	3940
YJM981	4012	4012	4012	4012	4012	4012	4012	4012	4012	4012

Table B.3: **Heuristic Testing of the *Seven Bridges Graph Genome* Pipeline Variant Calling Algorithm.** Number of variants called against the *Graph Genome Pipeline* reference graph for the simulated Bergstrom strains in Set 1 across 10 repeated runs. Runs where the number of variants called differ from the majority call have been highlighted in bold.

Strains	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9	Run 10
DBVPG1373	4801	4801	4801	4801	4801	4801	4801	4801	4801	4801
DBVPG6044	3112	3112	3112	3112	3112	3112	3112	3112	3112	3112
L1374	2878	2878	2878	2878	2878	2878	2878	2878	2878	2878
SK1	2986	2986	2986	2986	2986	2986	2986	2986	2986	2986
UWOPS83-787.3	2039	2039	2039	2039	2039	2039	2039	2039	2039	2039
W303	416	416	416	416	416	416	416	416	416	416
Y55	2102	2102	2102	2102	2102	2102	2102	2102	2102	2102
YJM978	4046	4046	4046	4046	4046	4046	4046	4046	4046	4046
YPS128	2588	2588	2588	2588	2588	2588	2588	2588	2588	2588
BC187	576	576	576	576	576	576	576	576	576	576
DBVPG1106	4741	**4742**	**4742**	4741	**4742**	4741	**4742**	4741	**4742**	4741
DBVPG1788	3700	**3698**	3700	3700	3700	**3698**	3700	3700	3700	3700
DBVPG6765	3742	**3743**	3742	3742	**3743**	**3743**	**3743**	3742	**3743**	3742
L1528	2986	2986	2986	2986	2986	2986	2986	2986	2986	2986
UWOPS03-461.4	3227	3227	3227	3227	3227	3227	3227	3227	3227	3227
UWOPS87-2421	2286	2286	2286	2286	2286	2286	2286	2286	2286	2286
Y12	2870	2870	2870	2870	2870	2870	2870	2870	2870	2870
YJM975	3994	3994	3994	3994	3994	3994	3994	3994	3994	3994
YJM981	4055	4055	4055	4055	4055	4055	4055	4055	4055	4055

Table B.4: **Heuristic Testing of the *BayesTyper* Variant Calling Algorithm.** Number of variants called using *BayesTyper* for the simulated Bergstrom strains in Set 1 across 10 repeated runs.

Strains	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9	Run 10
DBVPG1373	4971	4965	4975	4970	4973	4970	4969	4969	4972	4965
DBVPG6044	3295	3289	3288	3289	3292	3290	3290	3284	3290	3292
L1374	3007	3012	3011	3010	3005	3009	3005	3007	3013	3013
SK1	3124	3129	3121	3124	3129	3120	3131	3117	3127	3129
UWOPS83-787.3	2107	2109	2112	2111	2108	2110	2115	2116	2108	2115
W303	420	420	421	420	422	422	419	423	421	423
Y55	2290	2288	2288	2287	2286	2286	2289	2287	2286	2286
YJM978	4163	4155	4165	4165	4161	4157	4156	4167	4168	4160
YPS128	2703	2701	2700	2702	2698	2704	2697	2701	2699	2700
BC187	599	598	599	599	600	601	599	600	601	600
DBVPG1106	4935	4934	4939	4933	4937	4933	4931	4935	4941	4927
DBVPG1788	3832	3827	3831	3832	3808	3832	3829	3823	3822	3830
DBVPG6765	3908	3911	3908	3907	3886	3905	3909	3902	3909	3908
L1528	3115	3111	3108	3118	3109	3117	3107	3112	3115	3115
UWOPS03-461.4	3507	3509	3511	3507	3506	3509	3502	3503	3508	3503
UWOPS87-2421	2355	2354	2363	2359	2356	2360	2361	2358	2358	2357
Y12	3070	3070	3071	3060	3068	3069	3066	3067	3070	3069
YJM975	4115	4108	4119	4117	4109	4109	4107	4121	4121	4115
YJM981	4198	4196	4205	4204	4196	4195	4192	4208	4204	4201

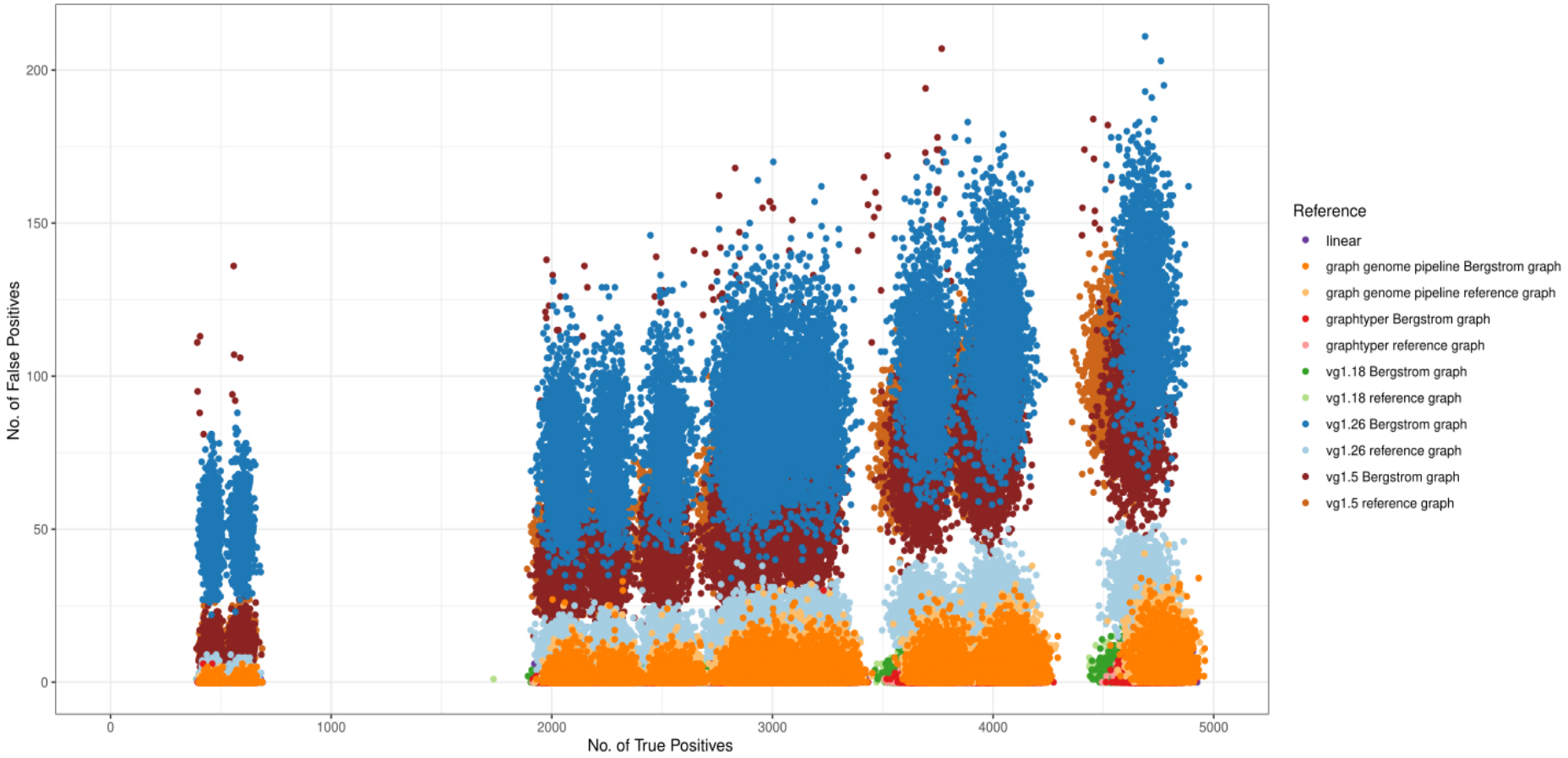


Figure B.19: **True Positive vs False Positive Variant Calls.** The scatter plot above depicts the no. of true positive and false positive variant calls from 11 different references (except *BayesTyper*) for 19,000 data sets.

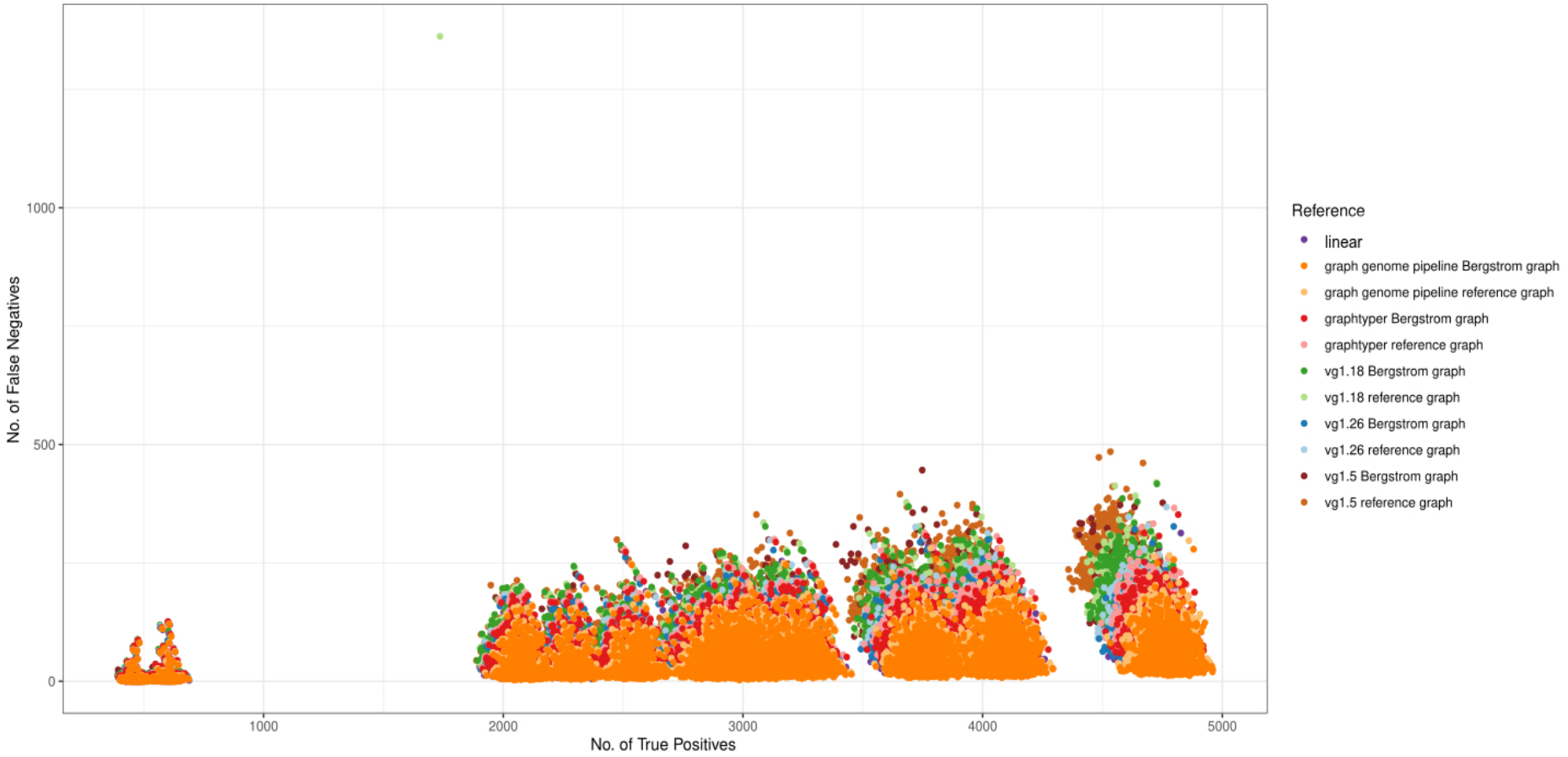


Figure B.20: **True Positive vs False Negative Variant Calls.** The scatter plot above depicts the no. of true positive and false negative variant calls from 11 different references (except *BayesTyper*) for 19,000 data sets.

Appendix C

Read Filtration

Reference Genome	Error Profile	Read Filter	Variant Classification	No. of Variants
1000 SNPs	Custom	Unanchored	True Positive	998
			False Positive	3364
			False Negative	2
		FAT-CIGAR 10	True Positive	996
			False Positive	1468
			False Negative	4
		FAT-CIGAR 20	True Positive	991
			False Positive	531
			False Negative	9
	Standard	Unanchored	True Positive	995
			False Positive	388
			False Negative	5
		FAT-CIGAR 10	True Positive	994
			False Positive	142
			False Negative	6
		FAT-CIGAR 20	True Positive	988
			False Positive	60
			False Negative	12
2000 SNPs	Custom	Unanchored	True Positive	1996
			False Positive	3905
			False Negative	4
		FAT-CIGAR 10	True Positive	1995
			False Positive	1221
			False Negative	5
		FAT-CIGAR 20	True Positive	1942
			False Positive	407
			False Negative	58
	Standard	Unanchored	True Positive	1996
			False Positive	368
			False Negative	4
		FAT-CIGAR 10	True Positive	1995
			False Positive	122

			False Negative	5
		FAT-CIGAR 20	True Positive	1946
			False Positive	32
			False Negative	54
3000 SNPs	Custom	Unanchored	True Positive	2980
			False Positive	3085
			False Negative	20
		FAT-CIGAR 10	True Positive	2975
			False Positive	1007
			False Negative	25
		FAT-CIGAR 20	True Positive	2781
			False Positive	236
			False Negative	219
	Standard	Unanchored	True Positive	2979
			False Positive	349
			False Negative	21
		FAT-CIGAR 10	True Positive	2976
			False Positive	91
			False Negative	24
		FAT-CIGAR 20	True Positive	2833
			False Positive	17
			False Negative	167

Table C.1: **Custom vs Standard Error Profiles.** Number of variants that were classified as true positives, false positives and false negatives for three simulated reference genomes containing 1,000, 2,000 and 3,000 SNPs when simulating sequence reads with the custom versus standard error profile in *pIRS*. The differences in variant classification without filtering (unanchored) and when filtered on the FAT-CIGAR string has also been shown. The 10 and 20 tags refers to the base length by which the reads were anchored against the reference genome.

Strains	Read Filter	Variant Type	TP	FP	FN
YPS128	Unanchored	SNPs	1769	27	543
		Indels	153	45	27
	CIGAR 10	SNPs	1768	24	544
		Indels	152	42	27
	CIGAR 20	SNPs	1768	22	544
		Indels	152	39	28
	CIGAR 30	SNPs	1766	19	545
		Indels	150	29	30
	FAT-CIGAR 10	SNPs	1764	10	548
		Indels	152	37	28
	FAT-CIGAR 20	SNPs	1723	7	589
		Indels	148	30	32
	FAT-CIGAR 30	SNPs	1399	4	913
		Indels	120	18	61
UWOPS03-461.4	Unanchored	SNPs	1173	18	357
		Indels	111	33	16
	CIGAR 10	SNPs	1173	16	357
		Indels	111	31	17
	CIGAR 20	SNPs	1173	15	357
		Indels	111	29	17
	CIGAR 30	SNPs	1172	13	358
		Indels	110	22	18
	FAT-CIGAR 10	SNPs	1171	9	359
		Indels	111	29	17
	FAT-CIGAR 20	SNPs	1158	6	372
		Indels	110	24	18
	FAT-CIGAR 30	SNPs	1013	4	518
		Indels	94	15	34
DBVPG6044	Unanchored	SNPs	1495	17	469
		Indels	143	39	23
	CIGAR 10	SNPs	1495	16	470
		Indels	143	38	23
	CIGAR 20	SNPs	1495	15	470
		Indels	143	35	23
	CIGAR 30	SNPs	1494	12	470
		Indels	141	25	25
	FAT-CIGAR 10	SNPs	1491	8	473
		Indels	143	34	23
	FAT-CIGAR 20	SNPs	1464	5	501
		Indels	140	28	26
	FAT-CIGAR 30	SNPs	1230	3	735

		Indels	119	16	47
Y12	Unanchored	SNPs	1443	20	450
		Indels	102	31	15
	CIGAR 10	SNPs	1443	19	450
		Indels	102	30	16
	CIGAR 20	SNPs	1443	16	451
		Indels	101	28	16
	CIGAR 30	SNPs	1443	14	451
		Indels	100	22	17
	FAT-CIGAR 10	SNPs	1440	8	453
		Indels	101	26	16
	FAT-CIGAR 20	SNPs	1416	6	478
		Indels	100	20	18
	FAT-CIGAR 30	SNPs	1208	4	686
		Indels	84	13	33
DBVPG6765	Unanchored	SNPs	1177	17	352
		Indels	110	34	17
	CIGAR 10	SNPs	1176	15	353
		Indels	110	33	17
	CIGAR 20	SNPs	1176	13	353
		Indels	110	29	18
	CIGAR 30	SNPs	1176	11	354
		Indels	109	22	19
	FAT-CIGAR 10	SNPs	1174	7	356
		Indels	110	29	17
	FAT-CIGAR 20	SNPs	1161	4	369
		Indels	109	24	19
	FAT-CIGAR 30	SNPs	1016	4	514
		Indels	93	14	35

Table C.2: **Read Filtering on Simulated Bergstrom Strains.** Average number of variants called from across 10 simulated datasets for each of the five strains. The numbers of SNP and indel variant calls have been shown for reads both with and without filtering on the CIGAR and FAT-CIGAR strings. TP stands for true positive variant calls, FP for false positive variant calls and FN for false negative variant calls.

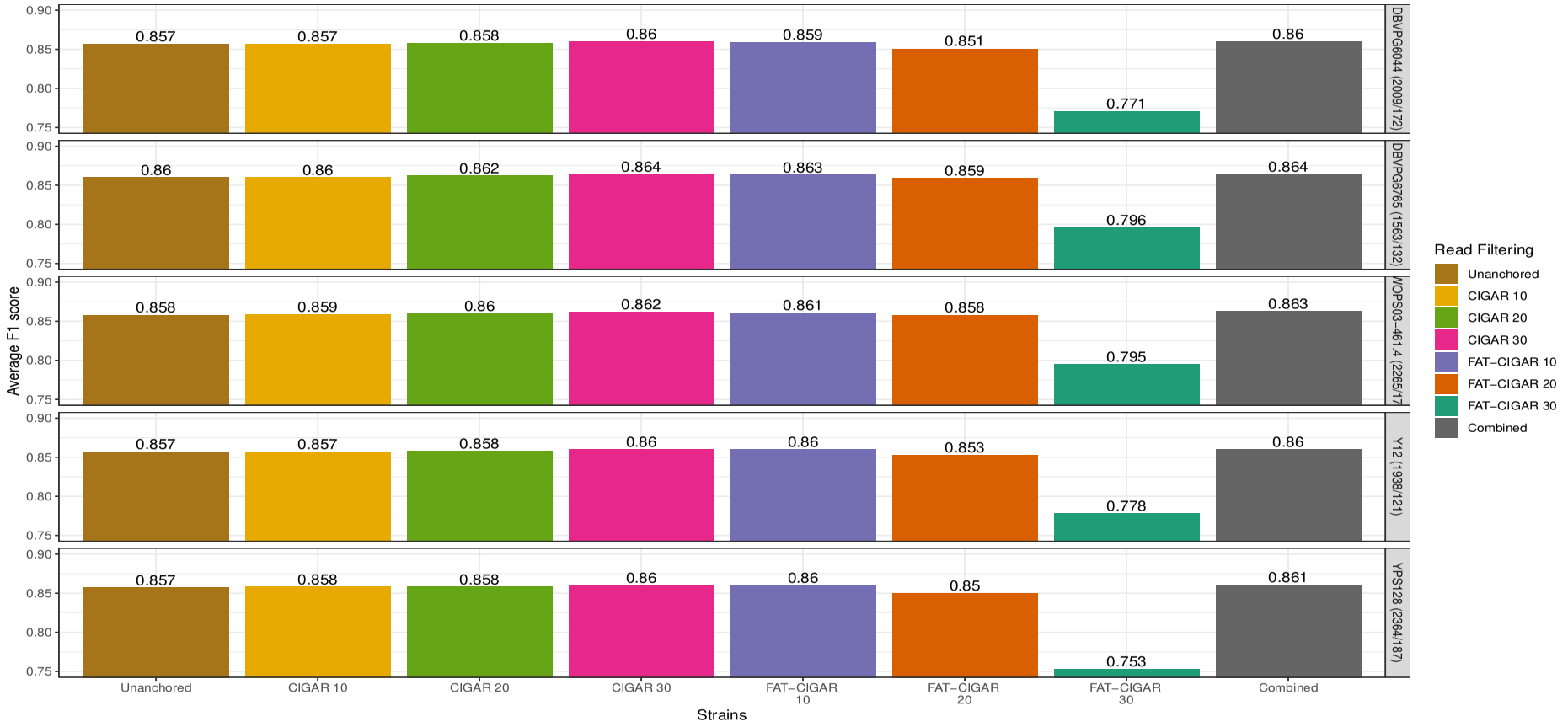


Figure C.1: **Bergstrom Strains Overall F1 Scores.** This bar chart shows the average overall F1 scores comparing the accuracy of variant calling after read filtering on both the CIGAR and FAT-CIGAR string for the five simulated Bergstrom strains. The combined bar (dark grey) refers to the F1 score from the combined SNP calls from reads filtered on the FAT-CIGAR string by 10 bases and indel calls from the reads filtered on the CIGAR string by 30 bases.