universität wien

DISSERTATION / DOCTORAL THESIS

Titel der Dissertation /Title of the Doctoral Thesis

Fast, accurate and user-friendly alignment of short and long read data with high mismatch rates

verfasst von / submitted by

Dipl.-Ing. Philipp Rescheneder, BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Doctor of Philosophy (PhD)

Wien, 2022 / Vienna 2022

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	UA 794 685 490
Dissertationsgebiet lt. Studienblatt / field of study as it appears on the student record sheet:	Molekulare Biologie
Betreut von / Supervisor:	Univ.-Prof. Dr. Arndt von Haeseler

Acknowledgments

I want to start by thanking my supervisor Arndt. Thank you for providing such a welcoming and engaging working environment that allowed me to focus on learning about science, and the fascinating field of molecular biology and genomics. Thank you for letting me explore my own interests and ideas and for the constant support when it came to smaller and bigger things like going to conferences, visiting other labs or starting collaborations or projects. I also want to thank Fritz, not only for the countless discussions about our shared work, but also for keeping to push me to go out of my comfort zone. This brings me to Mike who I want to thank for welcoming me in his group for a summer on very short notice and for giving me the opportunity to experience science in a different environment. This experience has had a profound impact on the course of my thesis as well as my career. I also want to thank Niko, Tobi, Nadia, Ivana and Meghan for all the great collaborations and Mihn for always being available for a chat about algorithms. Tina, Steffi, Olga, Ingo, Florian and Moritz - thank you for your support and the countless chats about science, work, politics, life and everything else. I have truly enjoyed the time with all off you and have learned a lot. Of course, also thank you to all the other members of CIBIV for the amazing working environment, the many great lunch breaks, and evenings we have spent together and last but not least teaching me the joy of drinking authentic Vietnamese green tea all day long. I also want to wholeheartedly thank my parents. Without your support I would not have been able to pursue my interests in the way I did. Also thank you to all my friends who have supported me throughout working on this thesis, especially Irene and Benni and lately Phill and Vania. Finally, I want to thank Tamara for being there for me and for supporting me to get this thesis over the finishing line.

Abstract

The advent of high-throughput sequencing has enabled us to study genomic variation at an unprecedented scale, providing us with insight into how genomes evolve, how phenotypes are influenced by genetic changes, and the mechanisms behind countless diseases. Large-scale projects, like the *1000 genomes project*, or similar projects for other model organisms sequenced thousands of genomes and cataloged the genetic variation they found. Most of these projects use a reference genome-based analysis approach where short high-quality sequencing reads are aligned to a high-quality reference genome like the human genome. Differences between the sequenced and the reference genome - mostly single nucleotide changes or small variants - are then detected using specialised tools.

Many analysis tools have been developed and optimised to efficiently analyse the immense amounts of data produced by these projects. However, these tools are often not applicable to experimental setups where either no high-quality reference genome exists, other less accurate sequencing technologies are used, more complex genetic variations are studied, or other sources of noise cause higher mismatch rates between the reads and the reference.

In this thesis we address this issue by introducing short and long read mapping tools that handle higher numbers of differences caused by sequencing error, evolutionary distance, or custom experimental designs, while offering the same ease of use and short runtimes as more specialised tools. Furthermore, we show how our analysis tools can enable researchers to study a wide range of genetic variations in model organisms as well as non-model organisms.

Parts of this thesis have been published in the following articles:

i) Sedlazeck, F. J.*, Rescheneder, P.*, Smolka, M., Fang, H., Nattestad, M., von Haeseler, A. and Schatz, M. C. (2018) Accurate detection of complex structural variations using single-molecule sequencing, *Nature Methods*, **15**(6), pp. 461468.

ii) Sedlazeck, F. J.*, Rescheneder, P.* and von Haeseler, A. (2013) NextGenMap: fast and accurate read mapping in highly polymorphic genomes, *Bioinformatics*, **29**(21), pp. 27902791.

* equal contributions

Other articles published during the course of this thesis:

i) Neumann, T., Herzog, A., V., Muhar, M., von Haeseler, A., Zuber, J., Ameres, S., L. and Rescheneder, P. (2019). Quantification of experimentally induced nucleotide conversions in high-throughput sequencing datasets, *BMC Bioinformatics*, **20**(1)

ii) Nattestad, M., Goodwin, S., Ng, K., Baslan, T., Sedlazeck, F. J., Rescheneder, P., Garvin, T., Fang, H., Gurtowski, J., Hutton, E., Tseng, E., Chin, C.-S., Beck, T., Sundaravadanam, Y., Kramer, M., Antoniou, E., McPherson, J. D., Hicks, J., McCombie, W. R. and Schatz, M. C. (2018) Complex rearrangements and oncogene amplifications revealed by long-read DNA and RNA sequencing of a breast cancer cell line, *Genome Research*, **28**(8), pp. 11261135.

iii) Muhar, M., Ebert, A., Neumann, T., Umkehrer, C., Jude, J., Wieshofer, C., Rescheneder, P., Lipp, J. J., Herzog, V. A., Reichholf, B., Cisneros, D. A., Hoffmann, T., Schlapansky, M. F., Bhat, P., von Haeseler, A., Kcher, T., Obenauf, A. C., Popow, J., Ameres, S. L. and Zuber, J. (2018) SLAM-seq defines direct gene-regulatory functions of the BRD4-MYC axis, *Science*, **360**(6390), pp. 800805.

iv) Drecktrah, D., Hall, L. S., Rescheneder, P., Lybecker, M. and Samuels, D. S. (2018) The Stringent Response-Regulated sRNA Transcriptome of Borrelia burgdorferi, *Frontiers in Cellular and Infection Microbiology*, **8**(JUL).

v) Herzog, V. A., Reichholf, B., Neumann, T., Rescheneder, P., Bhat, P., Burkard, T. R., Wlotzka, W., von Haeseler, A., Zuber, J. and Ameres, S. L. (2017) Thiol-linked alkylation of RNA to assess expression dynamics, *Nature Methods*, **14**(12), pp. 11981204.

vi) Sedlyarova, N., Rescheneder, P., Magn, A., Popitsch, N., Rziha, N., Bilusic, I., Epshtein, V., Zimmermann, B., Lybecker, M., Sedlyarov, V., Schroeder, R. and Nudler, E. (2017) Natural RNA Polymerase Aptamers Regulate Transcription in E . coli, *Molecular Cell*, **67**(1), p. 3043.e6.

vii) Popitsch, N., Bilusic, I., Rescheneder, P., Schroeder, R. and Lybecker, M. (2017) Temperature-dependent sRNA transcriptome of the Lyme disease spirochete, *BMC Genomics*, **18**(1), p. 28.

viii) Gesson, K., Rescheneder, P., Skoruppa, M. P., von Haeseler, A., Dechat, T. and Foisner, R. (2016) A-type lamins bind both hetero- and euchromatin, the latter being regulated by lamina-associated polypeptide 2 alpha, *Genome Research*, **26**(4), pp. 462473.

ix) Smolka, M., Rescheneder, P., Schatz, M. C., von Haeseler, A. and Sedlazeck, F. J. (2015) Teaser: Individualized benchmarking and optimization of read mapping results for NGS data, *Genome Biology*, **16**(1), p. 235.

x) Drecktrah, D., Lybecker, M., Popitsch, N., Rescheneder, P., Hall, L. S. and Samuels, D. S. (2015) The Borrelia burgdorferi RelA/SpoT Homolog and Stringent Response Regulate Survival in the Tick Vector and Global Gene Expression during Starvation, *PLOS Pathogens*, **11**(9), p. e1005160.

xi) Bilusic, I., Popitsch, N., Rescheneder, P., Schroeder, R. and Lybecker, M. (2014) Revisiting the coding potential of the E. coli genome through Hfq co-immunoprecipitation, *RNA Biology*, **11**(5), pp. 641654

Zusammenfassung

Das Aufkommen der Hochdurchsatz-Sequenzierung hat es uns ermöglicht genomische Variationen in einem noch nie dagewesenen Ausmaß zu studieren. Diese Untersuchungen gewährten uns Einblicke in die Entwicklung von Genomen, die Beeinflussung von Phänotypen durch genetische Veränderungen und die Mechanismen hinter zahlreichen Krankheiten. In groß angelegten Projekten wie dem *1000-Genome-Projekt* oder ähnlichen Projekten für andere Modellorganismen wurden tausende von Genomen sequenziert und die gefundenen genetischen Varianten katalogisiert. Die meisten dieser Projekte verwenden einen referenzgenombasierten Analyseansatz, bei dem kurze, qualitativ hochwertige Sequenzen ausgelesen (sequenziert) und mit einem hochwertigen Referenzgenom wie dem menschlichen Genom verglichen werden. Unterschiede zwischen dem sequenzierten Genom und dem Referenzgenom - meist einzelne Nukleotidveränderungen oder kleine Varianten - werden dann mit speziellen Analyseprogrammen detektiert.

In den letzten Jahren wurden zahlreiche hoch optimierte Analysewerkzeuge entwickelt um die immensen Datenmengen, die bei diesen Projekten anfallen, effizient zu analysieren. Diese Werkzeuge sind jedoch oft nicht auf Versuchsanordnungen anwendbar, bei denen entweder kein hochwertiges Referenzgenom existiert, andere, weniger genaue Sequenzierungstechnologien verwendet werden oder komplexere genetische Variationen untersucht werden und daher eine höhere Anzahl an Unterschieden zwischen dem sequenzierten Genom und dem Referenzgenome zu erwarten sind.

In dieser Arbeit befassen wir uns mit diesem Problem, indem wir Analysewerkzeuge für die effizienten Analyse von kurzen und langen DNA-Sequenzen vorstellen, welche eine höhere Toleranz für Sequenzierungsfehler und evolutionären Abstand

haben und dabei die gleiche Benutzerfreundlichkeit und kurze Laufzeiten bieten wie sie sonst nur höher spezialisierte Werkzeuge liefern. Darüber hinaus zeigen wir, wie unsere Analysewerkzeuge es Forschern ermöglichen, ein breites Spektrum an genetischen Variationen in Modellorganismen sowie in Nicht-Modellorganismen zu untersuchen.

Teile dieser Arbeit wurden in folgenden Artikeln publiziert:

i) Sedlazeck, F. J.*, Rescheneder, P.*, Smolka, M., Fang, H., Nattestad, M., von Haeseler, A. and Schatz, M. C. (2018) Accurate detection of complex structural variations using single-molecule sequencing, *Nature Methods*, **15**(6), pp. 461468.

ii) Sedlazeck, F. J.*, Rescheneder, P.* and von Haeseler, A. (2013) NextGenMap: fast and accurate read mapping in highly polymorphic genomes, *Bioinformatics*, **29**(21), pp. 27902791.

* equal contributions

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The development of DNA sequencing technologies has allowed us to read out genetic information as well as identify and catalogue differences ranging from single base changes to large structural rearrangements between single individuals or whole populations. The overall goal is to understand how genomic variation drives phenotypical changes, influences fitness, and causes diseases (Chong et al., 2015). A first major milestone was achieved in 2001 when Lander et al. and Craig Venter et al. finished the first drafts of a human reference genome. This project provided a foundation for studying human genetics. For the first time arbitrary human genomes could be independently sequenced and compared to the human reference genome to identify genetic variation (Pavlopoulos et al., 2013). By using the reference genome as a shared coordinate system, the identified variants could be stored in databases and shared with the research community, significantly accelerating research of human genetics. With the advent of second generation or high-throughput sequencing projects like the *1000 genomes project* set out to study and catalogue the full spectrum of genetic variation in multiple human populations (Gibbs et al., 2015). At the same time large genome-wide association studies were used to identify what variants or sets for variants are associated with a specific trait. With cost for sequencing decreasing exponentially and increasing throughput (Canzar and Salzberg, 2015), more large-scale projects like the *HapMap project*, the *10,000 Genomes Project*, the *ENCODE* project and the *Genotype-Tissue Expression project* (GTEx) further expanded our understanding of genomic variation and its functional consequences

in human populations. Similar projects were launched for microbes, plants, and other kingdoms (Pavlopoulos et al., 2013). Whilst different in goals most of these projects rely on shotgun sequencing at their core, meaning DNA is extracted, randomly fragmented, and subjected to DNA sequencing. The sequencing machine reads out (parts of the) DNA fragments. These sequences called *reads* are then compared to a reference genome to identify genomic variation (Pfeifer, 2017). The computational analysis of these experiments typically starts with aligning reads to a reference genome using a read aligner, followed by finding differences between reference and reads using a variant caller. Much research has gone into developing these tools. For example, more than 70 read aligners for short read sequencing have been developed (Canzar and Salzberg, 2015). Most initial large-scale sequencing projects were focused on human genomics or other well characterised model organisms. For these organisms high-quality reference sequences are available. Furthermore, polymorphism rates - meaning the number of differences between two individuals - are relatively low in organisms like human or mouse. As a result, when comparing reads from these experiments to their respective reference sequence only a small number of differences is expected. Furthermore, there has been a strong focus on short, mostly single nucleotide variants, in coding regions of the genome as these have historically been the easiest to identify and interpret with available technologies (Reuter et al., 2015). Regions with larger structural differences have mostly been ignored (Weischenfeldt et al., 2013). As a result, most read aligners have been optimised for this use-case. By optimising alignment algorithms to data sets with a low rate of differences between the reads and the reference, runtime and memory usage could be reduced by orders of magnitude, allowing read alignment to keep up with the exponential increase in available genomic data (Stephens et al., 2015).

However, since less focus has been put on non-model organisms or datasets with higher error or mismatch rates, tools available to analyse these data types are either too slow to deal with increasing data volume or require careful tuning and detailed knowledge of the data at hand. Furthermore, important classes of genetic variants like structural variations have been understudied or missed as they are difficult or impossible to detect in read alignments from state-of-the-art alignment tools. In

this thesis we address challenges in read mapping that arise off the beaten path of common workflows like short variant detection in model organisms. In the first chapter we propose and implement an algorithm for highly efficient short read mapping independent of error and mismatch rate. In the second chapter we extend this approach to noisy long third generation reads with a focus on achieving accurate read alignments even in the presences of large structural variations and throughout repetitive regions of the genome.

1.1 Genetic variation

In languages arbitrary amounts of information can be encoded by stringing together letters into sentences. In the case of the English language this alphabet would consist of 26 letters. Similarly, biological information is encoded by using a four-letter code consisting of the nucleotides: adenine, thymine, guanine, and cytosine. Through covalent bonds nucleotides can be arrange in long chains (Alberts B et al., 2002). The specific combination and order of the nucleotides in these polynucleotides encodes information similar to the letters in sentences. Through hydrogen bonds forming between adenine and thymine and guanine and cytosine two complementary copies of a polynucleotide can form a double helix structure referred to as Deoxyribonucleic acid (DNA). The vast majority of known organisms use DNA to encode their genetic information (Stefanie Tauber, 2013). As described by the central dogma of molecular biology, genes - functional units of information encoded in DNA - are transcribed into a single stranded RNA messenger molecule. Messenger RNAs are then translated into proteins (Crick, 1970). Proteins, one of the most important classes of biological molecules, carry out a majority of functions within cells and whole organisms, ranging from catalysing metabolic reactions, cell signalling, immune response to serving as structure components (Alberts B et al., 2002). Thus, they influence or define a wide range of the observable traits of an organism also called the phenotype. Through this process variations in the genetic makeup of an organism - the genotype - are translated into observable phenotypical changes. During cell division the two strands of a DNA molecule are separated.

Both strands then independently serve as templates for two newly synthesised complementary strands resulting in two identical copies of the initial DNA molecule. Especially higher organisms possess complex proofreading mechanisms, minimising errors during this process (Bębenek and Ziuzia-Graczyk, 2018). However, not all errors are detected or can be fixed. Thus, the two daughter cells might not have the exact same genetic information but contain small variations.
Genetic variation is generally categorised by the type of change and its size (Gibbs et al., 2015). Substitutions of a single nucleotide are called Single nucleotide variants (SNVs). When a small number of nucleotides is omitted or added the resulting variants are called small deletions or insertions (sometimes collectively referred to as indels). In contrast to these small variants, larger changes are called structural variations (SVs) sometimes also referred to as rearrangements. These three classes and their relevance to read mapping algorithms will be discussed in more detail below.

1.1.1 Single nucleotide variants and small indels

Single nucleotide variants are the simplest type of genetic variation. SNVs can either occur in regions of the genome that encode for a gene (coding regions) or regions that do not (non-coding regions). In coding regions, SNVs can alter the amino acid sequence of a protein (Robert and Pelletier, 2018). The terms SNV and Single nucleotide polymorphisms (SNPs) are often used interchangeably. A common distinction is that a SNP has to occur in a sufficiently large percentage of individuals in a population (e.g., $> 1\%$). Following this definition, all SNPs are SNVs but not the other way around. Small indel variants occur when either one or more bases are skipped (for example during replication) or when one or more bases are added. In genes, groups of three nucleotides (codons) code for a single amino acid. Thus, if the number of deleted or inserted bases is not a multiple of three, indels interrupt the so-called reading frame and change all codons downstream. Consequently, indels are more likely to have a strong negative impact on fitness causing indels to be rarer than SNVs (Lin et al., 2017).

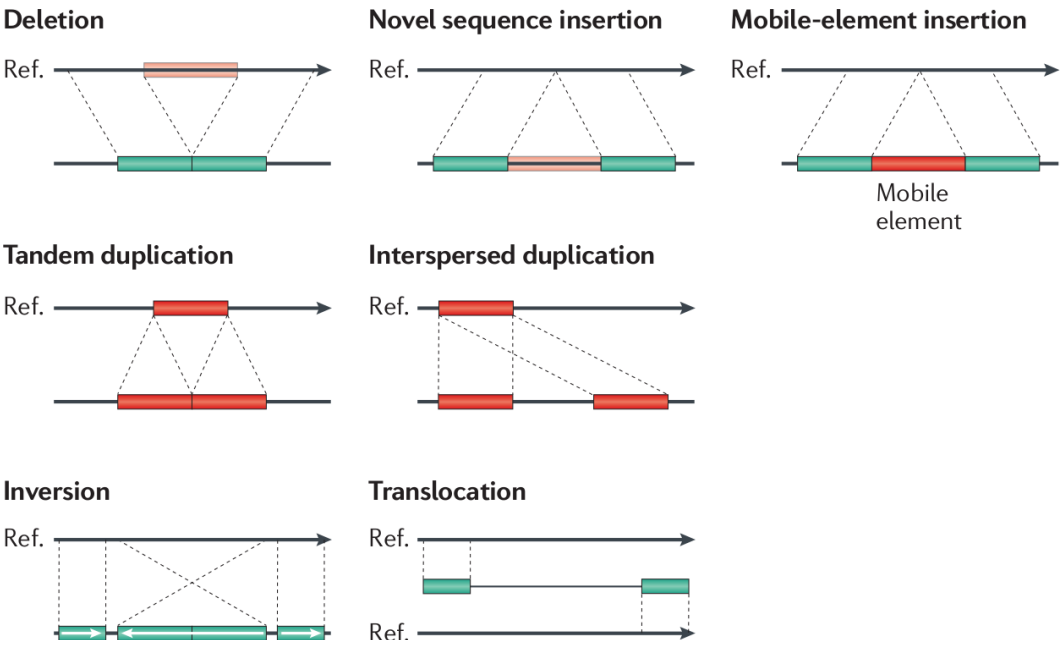


Figure 1.1: Different types of structural variation. Image taken from Alkan et al., 2011.

1.1.2 Structural variation

The second major class of variation is large alterations of the genome called structural variations. Initially, they were defined as variations larger than 1,000 bp. However, this threshold was mostly dictated by technical limitations (Alkan et al., 2011). With detection methods improving, the minimum size reduced over the last years. To date, a working definition of 50 bp and up has been established (Mills et al., 2011). SVs comprise several sub-categories including deletions, insertions, duplications, inversions, and translocations (Figure 1.1). Insertions can be further divided into novel sequence insertions and insertions of mobile elements like *Alu* or *L1* elements. Duplications might occur in tandem meaning the duplicated sequence is inserted directly up- or downstream of the template sequence or interspersed (Alkan et al., 2011). Furthermore, inversions and translocations are called balanced structural variations as they do not change the number of base pairs in the genome, whereas all others are called unbalanced. In humans, SVs make up the largest amount of differing base pairs (1-1.5%) when comparing the genome of two individuals (Mahmoud et al., 2019). SVs can arise during DNA recombi-

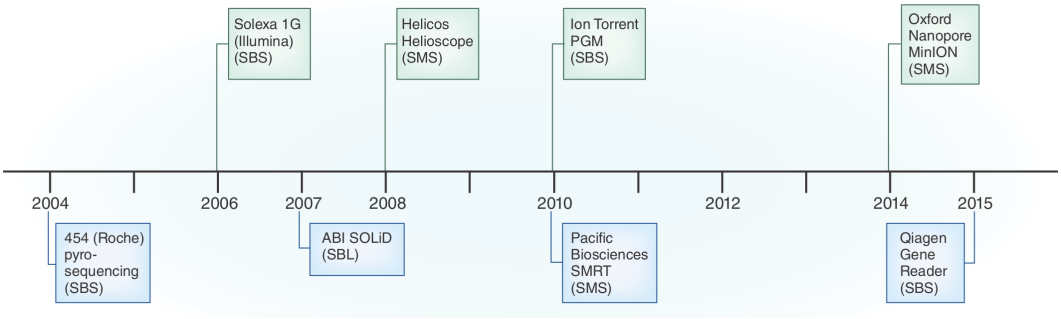


Figure 1.2: Timeline showing the year of introduction for major sequencing platforms since 2000. Image taken from Mardis, 2017.

nation, DNA replication and DNA repair. Errors in these processes are facilitated by the presence of low sequence complexity or repetitive regions (repeats) in the genome (Carvalho and Lupski, 2016). For example, during recombination two non-homologous repeats in the two sister chromatids can wrongly pair and cross over unequally (Schaeffer, 2016). Thus a larger percentage of SVs are found in repetitive or low complexity regions (Lucas Lledó and Cáceres, 2013).

1.2 High-throughput sequencing

Identifying and understanding genetic variation within a population requires sequencing of a large number of individuals. The introduction of Sanger sequencing, usually referred to as first generation sequencing, in the 1970ies revolutionised biological research (Mardis, 2017). For the first time researchers were able to investigate the genetic foundations of known phenotypic traits. Although Sanger sequencing was used to sequence the first draft of the human genome, outside of large collaborative projects the limited throughput and high cost mostly prohibited whole genome sequencing and population scale sequencing (Metzker, 2010). In the early 2000s multiple companies started commercialising new sequencing technologies that aimed at increasing sequencing throughput and decreasing cost (Mardis, 2017). Most notably Roche's 454 and Solexa's (now Illumina) Genome analyser. Unlike Sanger sequencing which yields a low number of longer reads (1000bp), these second-generation sequencing platforms aimed at providing a high number of

short (36-500bp) reads (Glenn, 2011). This was achieved by using an approach called *sequencing by synthesis* (SBS). Briefly, for SBS a high number of exact copies of a DNA molecule are produced first, using PCR also called clonal amplification (Reuter et al., 2015). Second, complementary strands are synthesised for all copies synchronously. Depending on the sequencing technology different mechanisms are used (for example fluorescent labelling of nucleotides) to monitor which nucleotide was inserted, during synthesis of the second strand. The key advantage is that this process can be run in parallel yielding millions or even billions of short reads in one sequencing run (Goodwin et al., 2016). As a result of this improved throughput the cost of sequencing dropped. For example, the cost of sequencing a human genome decreased from $\sim$ 100 million dollars in 2001 to around $1,000$ dollars in 2015[1]. This change allowed researchers to transition from mostly site or gene-specific assays to studying the whole genome and made population scale sequencing possible (Mardis, 2017). However, second-generation sequencing platforms come with limitations. For example, short read lengths make it impossible to study highly repetitive or low complexity parts of the genome and complicate identification of large genomic rearrangements (Mardis, 2017). Also, clonal amplification might introduce biases (Metzker, 2010). Third-generation sequencing platforms set out to solve these limitations by skipping amplification and sequencing single molecules. Producing strong enough signals from single molecules does not come without challenges either. As a result, error rates are generally higher than for second generation sequencing for most platforms. However, the read length is orders of magnitudes longer and sequence specific biases are reduced (Goodwin et al., 2016). Often the term next-generation sequencing technologies (NGS) is used to either refer to second only, or to second and third generation sequencing technologies. Furthermore, short read sequencing is now often used to refer to second generation sequencing and long read sequencing to refer to third generation sequencing. Today a mix of second and third generation sequencing technologies is used. Each of the platforms has their own strengths and limitations. In the following paragraphs we will briefly summaries the principles behind the three major sequencing platforms, *Illumina*, *Pacific Biosciences* and *Oxford Nanopore* and their

[1] https://www.genome.gov/about-genomics/fact-sheets/DNA-Sequencing-Costs-Data

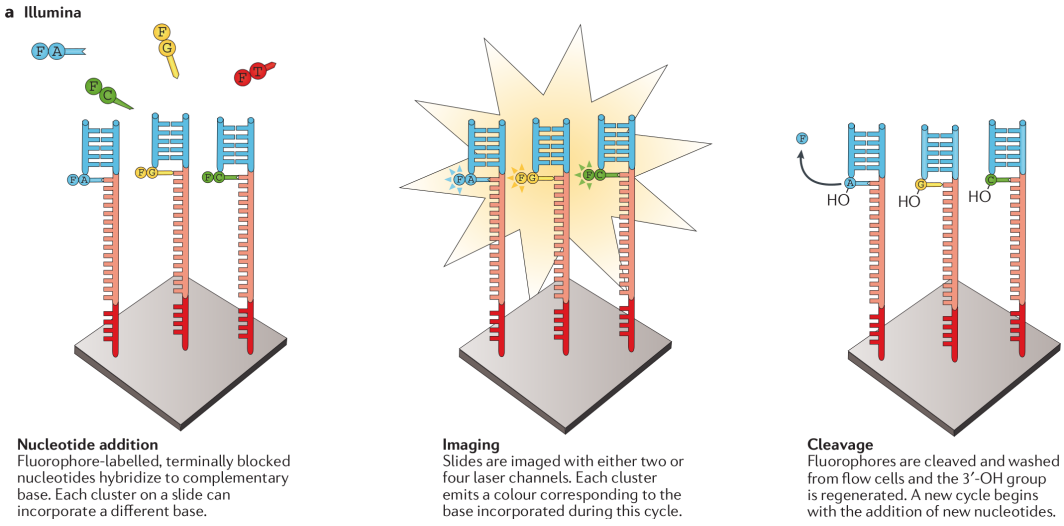


Figure 1.3: Schematic workflow of sequencing by synthesis. Image taken from Goodwin et al., 2016.

modes of error and biases. Understanding these characteristics is crucial for providing accurate and efficient computational analysis tools.

Independent of the sequencing platform, throughput has been improving rapidly over the last decade. In addition, NGS sequencing has been adopted in a wide range of research areas. Taken together, this caused exponential growth of sequencing data generation (Stephens et al., 2015) requiring computational analysis tools to not only provide accurate results but also to be extremely efficient.

1.2.1 Illumina short read sequencing

Before sequencing, extracted DNA is fragmented to a specific length. The fragment length is usually referred to as insert size (Goodwin et al., 2016). To prepare the fragmented molecules for sequencing, synthetic sequencing adapters are ligated to their 5' and 3'-end. This process is referred to as library preparation. The sequencing reaction itself takes place on a solid surface called flow cell and consists of the following steps:

Clonal amplification: During clonal amplification DNA fragments are immobilised on the flowcell and amplified into clusters of identical molecules (Goodwin

et al., 2016). To this end, flowcells come prepared with covalently bound forward and reverse primers. These primers are complementary to parts of the sequencing adapters allowing DNA fragments to base-pair with them. A polymerase then extends the primer by synthesising a strand of DNA complementary to the bound fragment. The now covalently bound fragments, from here on called templates, are amplified into clonal clusters through a process called bridge PCR. Briefly, the adapter on the unbound end of the fragments base pairs with reverse primers bound to the flow cell in close proximity. These primers are used to synthesis a complementary strand (Reuter et al., 2015). After denaturation of the template and complementary strand the process is repeated. Control of the number of initial fragments and PCR cycles ensures the formation of none overlapping clusters of identical copies of a single fragment.

Sequencing: The sequencing reaction is initiated by adding sequencing primers which are complementary to the sequencing adapters and a mixture of all four individually labelled and 3'-blocked deoxynucleoside triphosphate (dNTPs) (Goodwin et al., 2016). A polymerase binds the double stranded DNA region formed by the adapter and the primer and starts synthesizing a complementary strand based on the sequence of the template. However, since all dNTPs are 3'-blocked the polymerase will stall after incorporation of the first nucleotide. All unbound dNTPs are washed away, and a CCD camera is used to identify which dNTP was incorporated (Reuter et al., 2015). Finally, the fluorophore and the blocking group are cleaved from the incorporated dNTPs allowing the polymerase to incorporate the second nucleotide, initiating the second sequencing cycle. This process is done for all clonal clusters on the flowcell at the same time allowing for massively parallel sequencing.

Error profile

For this process to work it is required that all copies within a cluster are in synchronisation meaning exactly one nucleotide is added per sequencing cycle. Although efficiency of cleaving blocking groups as well as nucleotide incorporation rates are

high, they are not 100% (Metzker, 2010). Thus, over time more and more copies within a cluster can get out of sync. As a result, determining the correct bases for a cluster becomes harder over time, causing sequencing accuracy to decrease. This process is called dephasing and is one of the main factors limiting read length and sequencing quality (Metzker, 2010). Overall, sequencing error for Illumina reads is estimated to be between 0.1% and 1% (Nielsen et al. (2011), Goodwin et al. (2016), Canzar and Salzberg (2015)). Due to dephasing and other processes Illumina reads show the highest quality at the start of a read. Due to the blocking chemistry sequencing error is dominated by substitution errors as it is more like to misidentify the fluorescent signal than to either miss a base or add in an additional base. Due to amplification bias during clonal amplification Illumina sequencing might underrepresent AT- and GC-rich regions of the genome (Goodwin et al., 2016).

Read length

Early Illumina sequencing chemistries were limited to 25 bp single-end sequencing meaning that only one end of the DNA fragments could be sequenced. Today read lengths of up to 250 bp are supported (Goodwin et al., 2016). Furthermore, the addition of ”turn-around” chemistries enables sequencing of both ends of DNA fragments. Thus, a single cluster yields two reads instead of one. These two reads are guaranteed to originate from the exact same DNA fragment and are referred to as pair. Since paired-end sequencing allows reading twice the number of base pairs from a fragment it comes with advantages when aligning to repetitive regions of a genome.

1.2.2 Single molecule real time sequencing

Introduced in 2010, PacBio’s *Single molecule real time* (SMRT) sequencing was one of the first third generation sequencing platforms (Mardis, 2017). It allows for sequencing of long native DNA fragments without the need for clonal amplification or fixed chemical cycles (Goodwin et al., 2016). Instead of immobilizing DNA templates on a flowcell, a single polymerase is fixed to the bottom of a zeptoliter-sized

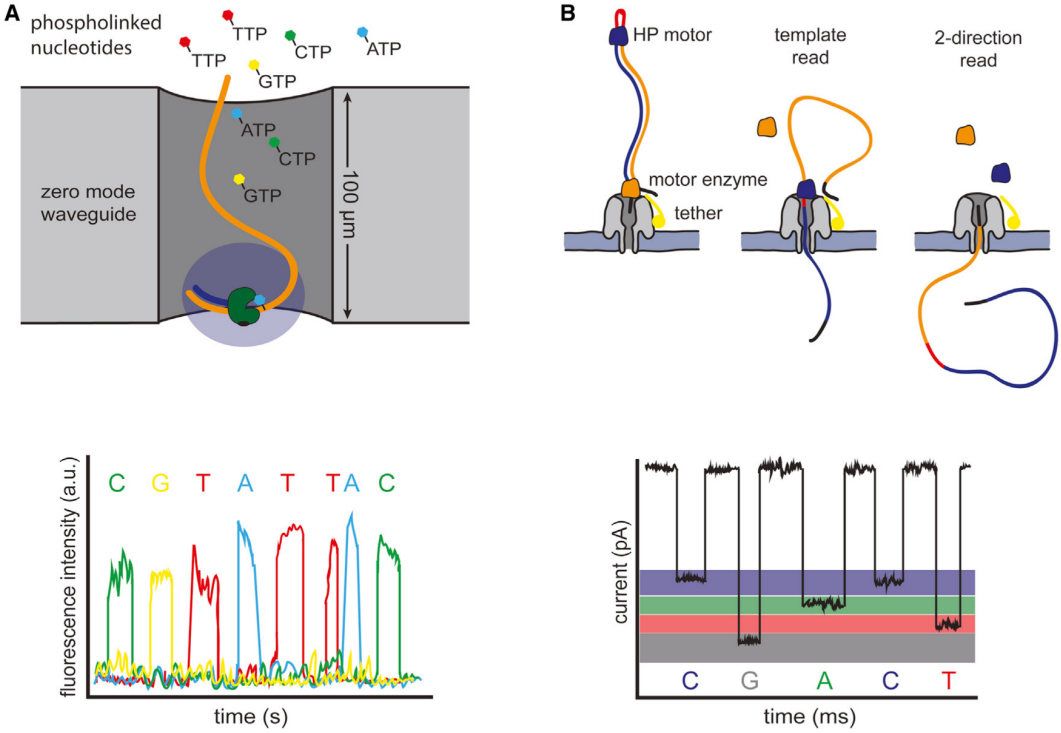


Figure 1.4: Illustration of (A) PacBio SMRT sequencing and (B) Oxford Nanopore sequencing. Image taken from Reuter et al., 2015.

well called zero-mode waveguide detector (ZMW) (Reuter et al., 2015). In addition to the polymerase, ZMWs contain dye-labelled nucleotides and the template DNA. As the polymerase copies the template, incorporation of dNTPs is continuously monitored using a laser/camera setup. In addition to the colour of the labelled dNTP also the duration of the incorporation is recorded. After incorporation, the polymerase removes the dye from the dNTP allowing for the incorporation of the next dNTP. This enables real time monitoring of the process and removes the need for blocking mechanisms.
During library preparation hair pin sequences are ligated to both ends of a double stranded DNA fragment. As a result, the molecules are circular meaning they can be sequenced multiple times by the same polymerase (Reuter et al., 2015). After sequencing has finished a high-quality consensus sequence can be computed from those copies. These reads are then called circular consensus sequencing (CCS) reads. In contrast, longer reads that stem from a single pass are called continuous long reads (CLR). It is up to the user which of these two modes to use. Since the same molecule is sequenced multiple times the overall throughput per SMRT cell is significantly lower when run in CCS mode. Also read length is reduced. However, read accuracy is significantly higher. CLR reads show an error rate of about 14% whereas CCS reads are typically more than 99% accurate (Goodwin et al., 2016). For both modes the main errors are short random insertion and deletions especially around homopolymers. A fundamental limitation in PacBio sequencing is that a single ZWM can only produce a single CLR or CSS read.

1.2.3 Oxford Nanopore sequencing

Unlike any other sequencing platform Oxford Nanopore sequencing does not rely on template DNA guided synthesis of a complementary strand (Goodwin et al., 2016). Instead, native DNA strands are sequenced directly by passing them through a protein pore. The pores are placed in a synthetic membrane and a specially designed motor protein ensures that the DNA translocates through the pore at a constant speed. Nucleotides passing through the pore create a characteristic change in current between both sides of the membrane. Measuring the current at 4,000 times

a second results in a so-called squiggle, a temporal recording of the current changes (Rang et al., 2018). The basecaller translates these squiggles into nucleotide sequences. Initially, DNA molecules were moving through the pore at a speed of 70 bases per second. However, through improvements in sequencing chemistry and electronics the speed increased to currently 450 bases per second (Rang et al., 2018). After a DNA molecule has passed through a pore, the motor protein disassociates from the pore allowing another DNA molecule to be sequenced by the same pore. As current changes can be monitored while DNA molecules move through the pore, Nanopore is the only sequencing platform that offers real-time sequencing. That means after starting a sequencing run, reads are immediately written to disk and available for downstream analysis. Similar to PacBio, Nanopore sequencing shows higher raw read error rates than short read sequencing. Early version of Nanopore flowcells were limited to a sequencing accuracy of roughly 60% (Rang et al., 2018). Through improvements in chemistry and software the current R9.41 pore shows a median raw read accuracy of 90 to 95% (Rang et al., 2018). Nanopore's main mode of error are short insertions and deletions with deletions in homopolymer regions being the most prevalent. These errors are mostly caused by nonuniform speed of the motor protein and problems with estimating the length of homopolymer regions. While rare reads of up to several megabases have been observed, read lengths of 10 to 50 kb are routinely obtained while so-called ultra-long read sequencing pushes read length into the range of 100kb and up (Rang et al., 2018).

1.3 Analysis of High-throughput sequencing data

Modern sequencing platforms are capable of producing billions of reads. Each read individually carries only relatively little information (Canzar and Salzberg, 2015). Thus, efficient analysis software is required to aggregate information contained in those small pieces and transform them into results that can be interpreted by researchers. There are two different workflows for analysing sequencing data: reference-based read mapping and de-novo assembly (Pavlopoulos et al., 2013).

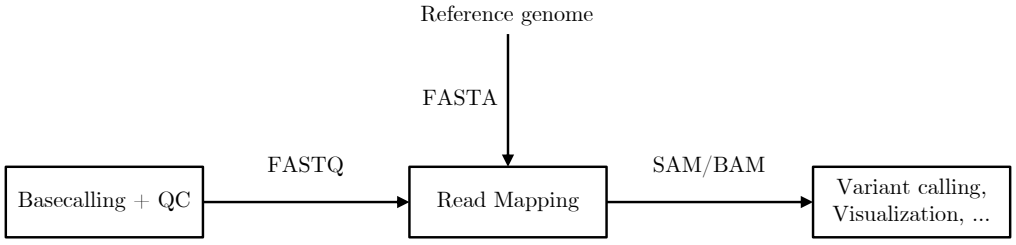


Figure 1.5: Basic reference-based read mapping workflow including most important file formats.

Reference-based read mapping uses prior knowledge in form of a reference sequence whereas de-novo assembly only relies on data produced by the experiment at hand. Conceptually, for de-novo assembly sequencing, reads are compared and collapsed if they show sufficient overlap. Under optimal circumstances, this allows a step-by-step reconstruction of the genome that was sequenced. Since no prior knowledge was used, this reconstruction is unbiased. However, genome assembly is hampered by repetitive regions in the genome where read overlaps cannot be uniquely resolved (Pavlopoulos et al., 2013). Furthermore, finding overlaps between all reads of a dataset is computationally intensive for long reads and not practical for short reads. Many data structures like bruijn graphs and algorithms that circumvent all vs all comparisons have been devised to optimise genome assembly from long and short reads. A comparison is outside of the scope of this work and readers are redirected to Sohn and Nam (2018) and Wick and Holt (2019) for an in-depth analysis. Independent of the approach used, genome assembly remains computationally expensive and heavily impacted by genome structure, ploidy, and read length. Thus, a reference-based approach has been the method of choice for most large-scale (re)sequencing and variant calling projects.

```
Reference:      CATCCC-GAATGAATGAAAAGCGTTTCCTAGTGGCTGTAATTTGCATCT-CCTGGTGGCTGA
                   || ||||.||||||||||   |||||||||||||.|||||||| |||||
Read:              CCGGAATCAATGAAAAGCG---CCTAGTGGCTGTAACTTGCATCTCCCTGG
```

Figure 1.6: Example of a read alignment. Matches are marked by the pipe symbol, mismatches by dots. Bases missing in the read but not the reference are called deletions. Additional bases in the read are called insertions.

1.3.1 Reference-based read mapping analysis

The inputs to a reference-based read mapping analysis workflow (Figure 1.5) are sequencing reads stored in a (compressed) FASTQ file and the sequence of the reference genome stored in a *FASTA* files (Pavlopoulos et al., 2013). FASTQ is a simple text-based file format that holds the nucleotide sequence for each read plus their respective per-base quality scores. These quality scores are expressed as PHRED scores and are a measure of the quality of every sequenced nucleotide. The FASTQ file and the sequence of the reference genome are the input for read mapping tools. The aligned reads are stored using the SAM/BAM format and are the input for all downstream analysis. After all reads have been aligned, variant calling tools use the read alignments of all reads to identify true biological variation and distinguish them from sequencing error (Koboldt, 2020).

Read mapping

Read mapping describes the process of identifying the most likely region of origin for a given read in a given reference genome (Canzar and Salzberg, 2015). This is done by finding the region of the reference genome that shows the highest sequence similarity to the given read. In addition to the position on the reference genome

read mapping will provide an associated (optimal) alignment between the read and the reference. This alignment establishes a position-wise correspondence between each nucleotide in the read and the reference as illustrated in Figure 1.6 (Canzar and Salzberg, 2015). All nucleotides in a read that match the reference are referred to as matches. Nucleotides that are different to the reference are called mismatches. These mismatches might indicate e.g., a SNV. Nucleotides that have either been deleted from or added to the sequenced genome are illustrated by dashes (Figure 1.6) and called indels.
Various characteristics of genome structure, genetic variation and sequencing platforms make read mapping more difficult:

Sequencing error As discussed in an earlier paragraph, different technologies have different sequencing error profiles. The most common sequencing errors in Illumina short reads are substitutions (Goodwin et al., 2016). Error rates differ between instruments but are reported to be in the range of 0.1 - 1.0 % (Canzar and Salzberg, 2015, Nielsen et al., 2011). Long read platforms on the other hand show between 5 - 20 % sequencing error mostly consisting of indel error (Mahmoud et al., 2019).

Quality and availability of suitable reference genomes Outside of established model organisms high-quality reference genomes are often not available (Rhie et al., 2021). Errors in the reference genome like shortened tandem repeats (Yang et al., 2019) cause differences between reads and the reference. Furthermore, for organisms or strains that have never been studied before reference genomes are not available. Thus, reads must be mapped to the evolutionary closest available reference genome. In such a case, the number of differences between reads and the reference will be determined by the evolutionary distances between the sequenced genome and the reference genome.

True biological variation Even if a high-quality reference genome exists the sequenced genome will never match the reference genome perfectly (Reinert et al., 2015). True biological variation within populations will lead to differences between the reads and the reference. In humans this has been estimated to be between 0.1% and 0.3% (Canzar and Salzberg, 2015, Mahmoud et al.,

2019). However, much higher variability is observed in other organisms. Furthermore, variation is not uniformly distributed throughout the genomes. Some regions like for example the MHC locus in humans is more variable than other highly conserved loci (Reinert et al., 2015). As discussed earlier, finding true biological variation is the main goal of many (re)sequencing projects. Thus, accurate mapping of reads which span variants is one of the main objectives of read mapping tools (Nielsen et al., 2011).

Repeats A common characteristic of genomes is that they often contain one or more copies of identical or near identical sequences or motifs. These repeated sequences (repeats) are frequently caused by duplication events, transportable elements, or satellite repeats in centromeric or telomeric regions (Cechova, 2021). For the human genome an estimated 50 - 69 % of all nucleotides can be characterised as being part of a repeat (de Koning et al., 2011). Repeats longer than the read length will result in so-called multi-mapping reads. For these reads it is impossible to assign a single unique region of origin.

Read mapping tools need to account for these sources of sequence differences and biases to allow for accurate identification of the regions in the genome that show the highest sequence similarity to a given read.
A common approach to determine pairwise sequence similarity and compute sequence alignments are dynamic programming algorithms like the *Needleman-Wunsch* (NW) or the *Smith-Waterman* algorithm (SW).

Sequence alignment algorithms: Sequence alignment algorithms require a *scoring function* that assigns a positive score to each match and a negative score (penalty) to each mismatch and indel. For two sequences S and Q, the global alignment algorithm (also referred to as Needleman-Wunsch algorithm) uses *dynamic programming* to find the optimal sequence alignment given the respective scoring function in quadratic time and space complexity (Knuth, 1997). Conceptually, the algorithm works by filling in a two-dimensional matrix V of length m times n where m and n are the length of S and Q, respectively. Each element $V_{i,j}$ of the matrix represents the optimal alignment score - i.e. the sum of all individual match, mismatch and indel scores of the optimal alignment - between the sub-sequences $S_{0..i}$ and $Q_{0..j}$.

Filling in the first row and column is trivial. All remaining cells can be filled in using a simple *recurrence*:

$$V_{i,j} = \max \begin{cases} V_{i-1,j-1} + s(S_i, Q_j) \\ V_{i-1,j} + G \\ V_{i,j-1} + G \end{cases}$$

, where s is the scoring function and G is the penalty for introducing a one base pair insertion or deletion. In other words, the value of each cell only depends on the value of their three neighbouring cells. After filling in V, the bottom and left most cell (Vi, j) contains the optimal alignment score for S and Q. This part of the algorithm is also called the *forward step*. In a second step (called *backtracking*), the optimal sequence alignment is reconstructed using the values in V. First, it is determined from which of the cells the score in $V_{i,j}$ was derived from. If it was $V_{i-1,j-1}$ the reconstructed alignment will contain a match or mismatch (depending on the sequence of S and Q). If it was $V_{i-1,j}$ or $V_{i,j-1}$ the alignment will contain an insertion or deletion, respectively. Repeating this step until reaching $V_{0,0}$ will reconstruct the optimal alignment between S and Q. A variation of this algorithm called Smith-Waterman algorithm computes so called *local alignments*. Instead of the best alignment between S and Q it will find the alignment with the highest score for all possible sub-strings of S and all sub-strings of Q.

If we think of S as our reference genome and of Q as our read, it becomes obvious how the SW algorithm could be used to find the optimal read mapping. Factors like the number of differences between the read and the reference genome and the repeat content of the reference genome have little to no influence on this approach. Furthermore, alignment algorithms can be extended to model indels of multiple consecutive base pairs stemming from a single mutation event (Canzar and Salzberg, 2015) using gap costs. As introduced earlier indels are visualised by using dash characters in alignments. We can think of a gap as a consecutive run of one or more dash characters in an alignment (Knuth, 1997) (see Figure 1.6). For a naive alignment algorithm, the penalty for a gap of length n equals n times the penalty for a gap of length one (linear gap costs). Evidently, when using linear gap costs two

gaps of length two are assigned the same penalty as one gap of length four. For biological variation it is however more likely to observe one longer gap than multiple shorter gaps in close proximity. Thus, so-called affine gap costs have been introduced which use a separate gap-open g_O and gap-extension penalty g_E and define the cost of a gap with length n as:

$$G(n) = g_O + n * g_E$$

Affine gap costs have been shown to model genetic variation more accurately and thus improve alignments of reads spanning indels (Knuth, 1997).
The main disadvantage of alignment algorithms is that their runtime and memory requirements scale with the size of the reference genome. As a result, using sequence alignment algorithms alone for read mapping is impractical even for smaller bacterial genomes and computational infeasible for larger mammalian or plant genomes (Canzar and Salzberg, 2015). Note, optimised alignment algorithms exist which reduce either runtime or memory complexity. However, even these algorithms are not efficient enough to be used for read mapping directly.

String matching: Searching large databases of text for exact matching substrings is an important and well-studied problem in computer science (Knuth, 1997). Most notably *indexed string matching* allows for finding occurrences of a search pattern without having to scan the full database (Canzar and Salzberg, 2015). *Suffix trees* or the more compact *Suffix arrays* allow for search times that only scale with the length of the pattern (the read sequence in our case) and not the reference genome. Although, the size of a suffix array scales only linearly with the input text (the reference genome), the amount of memory required still makes suffix arrays impractical for routine use in read mapping tools (Canzar and Salzberg, 2015). Only after Ferragina and Manzini had shown how to reduce memory requirements by using the *Burrows Wheeler Transform* (BWT) when presenting their FM-Index, string matching started to be widely applied to the read mapping problem. BWT based data-structures like the FM-Index mimic *suffix/prefix tries* but combine them with compression. This makes them similar in size to the original text, while maintain-

ing near optimal search time (Canzar and Salzberg, 2015). Note, the FM-Index still only permits searching for exact matches and thus cannot be applied to read mapping without modifications.
Most read mapping tools use heuristics to either speed up sequence alignments, extend BWT/FM-Index like data-structures to account for sequencing error, or combine the two approaches to enable fast, efficient, and accurate mapping. Since read mapping is one of the first steps of a reference-based analysis workflow, the choice of reads mapper and its algorithmic approach can influence downstream analysis (Cechova, 2021). In the following we will briefly discuss the trade-offs between different read mapping algorithms for short and long read mapping.

Short read mapping

As Illumina's sequencing platform has been dominant for more than a decade[2] most short read mapping tools have been optimised for aligning billions of reads with a length between 25 and 250 bp, containing mostly substitution error, to an arbitrary high-quality reference genome. As discussed earlier, it is computationally infeasible to align these reads using pairwise sequence alignment algorithms alone. However, many heuristics have been introduced to reduce runtime and memory usage while maintaining the favourable characteristics of sequence alignment algorithms. The key observation was that the vast majority of the reference genome will share little to no sequence similarity with a given read (Canzar and Salzberg, 2015). Another observation was that reference genomes, for all practical purposes, are static. Thus, data structures that allow for a quick identification of all regions in the reference that show a sufficiently high sequence similarity to a read are beneficial, even if they are compute- or memory-intensive to build as they can be reused over and over again. As a result, many approaches have been developed that follow a so-called *seed and extend* or *seed and vote* approach (Canzar and Salzberg, 2015). Briefly, in the seeding phase an index is used to quickly identify short exact matches between a read and the reference (seeds). Most commonly the index is a hash table that stores the positions in the reference genome of all k-mers with a fixed k (e.g., 13).

[2] https://frontlinegenomics.com/how-did-illumina-monopolize-the-sequencing-market/

As hash table look-ups only take constant time, retrieving reference locations for all k-mers occurring in a given read is fast and only scales with the read length. In the extension phase seeds in close proximity to each other are connected using a more time intensive sequence alignment algorithm. Seed and vote algorithms work in a similar fashion. However, they use the number of seed matches in close proximity as a proxy for the overall similarity between the read and the reference (Canzar and Salzberg, 2015). During an additional filtering step all regions with no or an insufficient number of seed matches are discarded. All remaining regions are "cut out" from the reference and subjected to a pairwise sequence alignment with the read sequence. Different approaches for how to implement the seeding and filtering steps have been proposed and were summarised by Canzar and Salzberg (2015).
With the introduction of the FM-Index more read mapping tools started adopting a mapping strategy that does not involve dynamic programming-based sequence alignment algorithms. A straightforward approach to apply the BWT based FM-Index to the read mapping problem is to assume an upper bound for the number of allowed differences k. A given read can then be mutated into all possible sequences with a maximum of k differences and the BWT index searched for all these strings. While this strategy can be implemented using a straightforward recursive function the search time increases exponentially with k. Tools like BWA (Li and Durbin, 2009) and Bowtie (Langmead et al., 2009) use a similar approach but implement pruning strategies and heuristics to allow for fast and memory efficient mapping of short reads ($<$100bp). Tools like bowtie2 (Langmead and Salzberg, 2012), in contrast, rely on a seed and extend approach but use a modified FM-Index for identifying fixed length seeds. The advantage is that here k can be low (e.g., 1) allowing for fast and memory efficient seed identification even in repetitive regions. We will discuss differences in runtime and mapping accuracy for specific mapping strategies in more detail in chapter 3.

Long read mapping

Long reads come with 5-20% sequencing error. Furthermore, reads are on average two orders of magnitudes longer than short reads. Thus, short read mapping

tools are generally not applicable to long read data. However, similar to short read mapping tools, current long read mappers like BLASR (Chaisson and Tesler, 2012) detect short exact matches between the read and the reference first. Due to the high error rate this step alone is not informative enough to determine the most likely mapping location of the full read. Thus, long read mappers add a clustering step - often called chaining - to combine short matches in close proximity which are consistent with an alignment of the full read, discarding spurious matches to unrelated parts of the genome (Canzar and Salzberg, 2015). Finally, they use sequence alignment algorithms to either compute the final alignment between the read and the reference (BLASR) or connect already identified exact matches (BWA-MEM (Li, 2013)). BLASR uses either a suffix array or a FM-index for the first step while BWA-MEM uses an algorithm to identify so-called super maximal exact matches between the read and the reference (Liu and Schmidt, 2012). This algorithm identifies all unique matches i.e., non-overlapping exact matches between the read and the reference that cannot be extended any further (Canzar and Salzberg, 2015). Due to the read length multi-mapping reads are observed less frequently with long reads. However, split reads - reads that cannot be mapped to the reference with a single linear alignment - are commonly observed for reads that overlap structural variations and need to be addressed properly.

1.3.2 Variant calling

Variant calling is the process of identifying or *calling* genetic variation from a sequencing dataset (Olson et al., 2015). Typically, calling of SNV and small indels (up to 30bp) is carried out by SNV calling tools (often called SNP callers). Larger structural variations are called using specialised SV calling tools (*SV callers*). Accurate alignments are crucial for variant detection as alignment errors could be misinterpreted as true biological variation (Nielsen et al., 2011). In the following we will give a brief overview of SNV and SV calling and their respective challenges they pose for read mapping.

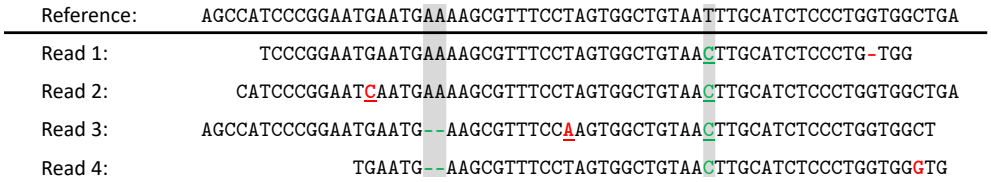


Figure 1.7: Example pileup of five reads against a reference. Sequencing error is coloured in red and genuine biological variation in green.

Single nucleotide variation calling

SNV callers screen read alignments for sites where one or more reads differ from the reference genome (Nielsen et al., 2011). These differences can either be caused by sequencing error or true variation in the sequenced individual. The process of SNV calling can be visualised by piling up all the reads that overlap with a given region of the reference as shown in Figure 1.7. Each column of this pileup represents a single nucleotide in the reference (for simplicity we ignore insertions here). As long as sequencing errors (indicated by the red letters in Figure 1.7) is mostly random, the chance for observing the same error multiple times in one column is low. In contrast, for true variation we expect roughly 50 % of the reads to show the same alternate nucleotide for a heterozygous variant. For a homozygous variant, we expect most of the reads to show an alternative nucleotide (i.e., not the nucleotide present in the reference genome).

The first generation of SNV callers simply scanned the pileup column by column. If in a given column the number of reads that show an alternative allele exceeds a certain fixed threshold (e.g., 20% of the overall read depth), a SNV or indel is called (Nielsen et al., 2011). Typically, a second fixed threshold (e.g., 80% of the overall read depth) was used to determine whether the sequenced individual is heterozygous or homozygous for the variant. This process is also called genotyping. However, for samples with lower read depth this approach might lead to undercalling of heterozygous variants (Nielsen et al., 2011). In addition, it doesn't take into account non-random sequencing errors and does not provide reliable quality scores along with the variant calls. For this reason, later SNV callers use several probabilistic methods to included base quality scores, models of sequencing error and information about allele frequencies or patterns of linkage disequilibrium when

calling variants from a large number of samples (Nielsen et al., 2011).
Independent of the SNV calling algorithm, accurately aligned reads are required for high-quality SNV calls (Nielsen et al., 2011). In the context of short reads, it is most important to identify the correct mapping location for reads overlapping repeats as well as highly variable regions of a genome. Reads that cannot be mapped uniquely must be reliably identified as multi mapping reads and assigned a low mapping quality score. Otherwise, reads incorrectly mapped to a similar but not identical repeat could cause false positive SNV calls. Furthermore, read alignments need to be accurate around short indels to avoid alignment artifacts being called as true variation.

Structural variation calling

Multiple strategies for SV calling from short read data exist (see Figure 1.8).

Read depth So-called copy number variation (CNV) callers use read depth to detect large ($>$ 1000 nucleotides) deletions or duplications. A region that was duplicated in the sequenced individual will yield roughly double the read depth when mapping to a reference that does not have the duplication (see Figure 1.8 second column). Equally, deleted regions will yield half the read depth for heterozygous deletions or no reads at all for homozygous deletions. Detecting these changes allows for robust CNV calling. Evidently, this approach is not applicable to balanced SVs like translocations or inversions. Furthermore, depth based CNV callers cannot accurately detected break points (Alkan et al., 2011).

Read pair For paired-end sequencing read-pair information can be used for SV calling. Two reads that form a pair are expected to map within a certain distance (the insert size). Furthermore, their direction is determined by the library preparation. If a read pair is mapped across a break point of a SV either the distance or the orientation will be affected. For example, the distance between two reads from a read pair spanning the break point of a 10 kbp deletion will be much larger than expected when mapped to the reference. While

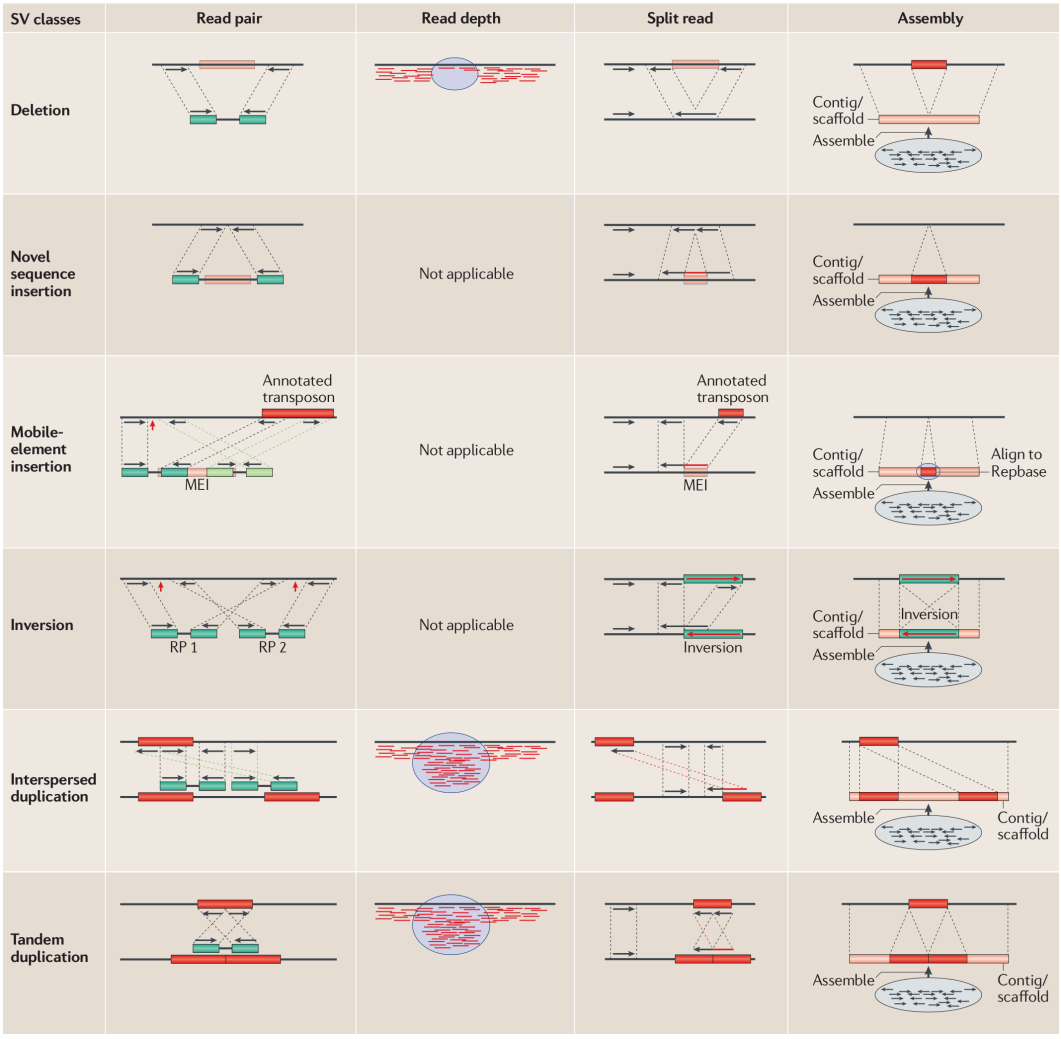


Figure 1.8: Different strategies for calling SVs from sequencing data. Image taken from Alkan et al., 2011.

this approach is able to identify all types of SVs, reliable detection of exact break points is not possible (Alkan et al., 2011).

Split read To detect exact break points a SV caller must take split read alignments into account (Alkan et al., 2011). Reads that span a breakpoint of a large SV will need to be split into two separate alignments when aligning them back to the reference. In the case of a 10 kbp deletion the part of the read that is upstream of the break point will for example map 10 kbp upstream of the second part of the read. Thus, accurate split read alignments can be used to identify exact break points for SVs (Alkan et al., 2011).

Another approach for detecting SVs is to create a de-novo assembly from the reads first and call SVs from a whole genome alignment between the assembly and reference genome. While this strategy can help to identify complex SVs, it is more compute intensive and lacks sensitivity when calling heterozygous SVs (Nattestad, 2017).

As discussed in a previous paragraph, a characteristic of SVs is that they mostly occur in repetitive regions of the genome. Since short reads can often not be aligned uniquely in these regions, SV calling from short reads is inherently difficult (Mahmoud et al., 2019). Furthermore, short reads are less likely to fully span a SV thus complicating SV calling even further. Long reads, on the other hand, span most repetitive regions and have a higher chance to completely span SVs like large insertions. There are two ways a SV can be represented in long read alignments. Insertions or duplications with lengths exceeding the read length as well as inversions and translocation are represented as split alignments. Insertions, deletions, and duplications with lengths significantly shorter than the read length can either be represented as split alignments or as a single alignment. For example, a 1,000 bp deletion spanned by a 5,000 bp read could be expressed as two 2,500 bp alignments mapping 1,000 bp apart or as a single 6,000 bp alignment containing the full 1,000 bp deletion. To enable accurate SV calling, long read mapping tools must be able to create accurate split read alignments and alignments fully spanning SVs in the presence of the high indel based sequencing error observed in long read sequencing technologies.

Chapter 2

Main contributions of this thesis

While there is a wealth of read mapping tools available, we found that the majority of these tools, especially the most popular ones, only focus on a specific well-defined use-case. Namely, mapping of short high-quality reads to a high-quality reference genome. The goal of this thesis is to develop easy-to-use read mapping tools that map reads efficiently and accurately and enable accurate variation calling in the absence of a closely related high-quality reference sequence or in the presence of high levels of sequencing error.

In chapter 3 we present an easy-to-use short read mapping tool that can tolerate a high number of differences between the reads and the reference and therefore enables researchers to apply large scale NGS reference-based analysis outside of standard model organisms. Our main contributions were:

Optimised index data-structure Our index improves over available hash-tables by using domain knowledge like typical genomes sizes, useful ranges of k for read mapping and knowledge about the repeat content and structure of genomes to allow for efficient querying of k-mer positions while being compact enough to fit into main memory.

Adaptive seed filtering The vast majority of reads in a dataset share sufficient similarity with only a single region in the reference genome. The correct mapping location for these reads is easily determined in the seeding step and needs only minimal verification using sequence alignments. Only a small percent-

age of reads - mostly reads stemming from repetitive or highly polymorphic regions - are typically hard to map and thus need extensive verification using sensitive sequence alignments. We defined a seed filtering heuristic that efficiently discriminates between easy and hard to map reads in the seeding step enabling accurate and fast mapping at the same time.

Self-tuning of most important parameters All important read mapping parameters are automatically estimated from the input data. Thus, no custom parameters have to be adjusted by the user.

Furthermore, we found that existing long read mapping tools are unable to align noisy long reads accurately in the presence of larger structural variations. First, routinely used affine gap costs are unable to accurately align reads that span true biological insertions and deletions in the presence of sequencing error that is found in current single molecule sequencing platforms. Second, long reads that span breakpoints caused by large structural variation cannot be aligned in a single linear alignment but need to be split. The optimal strategy for splitting reads depends on the length of the reads and the length and type of SV. To address these issues in chapter 4 we introduce a new long read mapping tool that includes:

Convex gap costs We introduce a heuristic alignment algorithm that mimics convex gaps costs enabling accurate alignment of reads spanning large insertions and deletions even in the presence of sequencing error.

SV aware per-read selection of highest scoring non-overlapping alignments To improve SV calling we implemented a step-wise alignment approach were high confidence linear alignments are found first and then pieced together to create the optimal split read alignment.

Chapter 3

NextGenMap: Fast and accurate read mapping independent of evolutionary distance

3.1 Introduction

High throughput sequencing (HTS) technologies produce several million reads per run with read lengths ranging from 36 to >1000 bp. The increasing read length and the advent of technologies like Ion Torrent and MiSeq drive the need for new efficient read mapping methods. In addition, the demand increases for methods that can cope with high sequence divergence and at the same time are user friendly.
Two main groups of read mapping programs are distinguished based on their alignment methods (Nielsen et al., 2011). First, Burrows Wheeler transformation (BWT) based methods e.g. BWA (Li and Durbin, 2009), which are fast but optimized for short reads and genomes with low polymorphism. Second, hash based methods like Stampy (Lunter and Goodson, 2011), which are slow but also suited for highly polymorphic genomes. A method that combines speed and accuracy provides a quantitative and qualitative improvement of current methods. To this end, we introduce *NextGenMap*, a method that is faster than current BWT based methods and at the same time handles short and long reads independent of the number of differences

between read and reference genome. *NextGenMap* implements new techniques to efficiently identify a minimal set of genomic regions on the reference genome that share a sequence similarity with a read. Furthermore, *NextGenMap* estimates the most important parameters (e.g. minimal *k*-mer matches, alignment corridor width) to reliably map reads. To achieve a short run time, *NextGenMap* makes use of multi core CPUs and if available any OpenCL enabled graphic card (GPU) independent of its manufacturer. Therefore, *NextGenMap* can bridge the gap between fast and flexible mapping programs. *NextGenMap* supports fasta, fastq, SAM, and BAM as input formats; and outputs SAM and BAM files. Furthermore, *NextGenMap* maps single-end and paired-end data, offers a local (Smith Waterman) and an end-to-end alignment mode and supports aligning bisulfite treated reads (Dinh et al., 2012). The peak memory consumption of *NextGenMap* ranged from 5 to 6 GB depending on the read length. Thus, *NextGenMap* runs on a standard desktop computer with 8 GB of RAM.

3.2 Methods

NextGenMap comprises three steps. First, it splits the reference genome into overlapping *k*-mers and stores the positions in a hash-table. Second, *NextGenMap* identifies the genomic regions, where a read potentially maps to. To this end, the *k*-mers from each read are extracted and putative genomic locations are retrieved from the hash-table. Only regions on the genome where the number of *k*-mer matches exceeds a certain threshold are considered as candidate mapping regions. Unlike other methods *NextGenMap* automatically determines a read specific threshold, rather than one threshold for all reads. Third, *NextGenMap* computes the alignment score for the candidate mapping regions. For the candidate region(s) with the highest alignment score the full alignment is computed and reported. This final step is performed using a extended implementation of the *MASon* library (Rescheneder et al., 2012).

Like other hash-based methods (Lunter and Goodson, 2011) we use overlapping *k*-mers (substrings of *k* letters that occur in the reference genome) to identify regions on the genome that show a high similarity to a read. To this end, we construct a

hash-table to store the *k*-mer positions in the reference genome. If a *k*-mer occurs in the reference genome and in a read, we call it seed-word for the particular read. We will use seed-word whenever it is clear that a particular read is meant. One or more seed-words in close vicinity define a candidate-mapping region (CMR) in the genome, where the read potentially maps. Thus, a read may have many CMRs. For each CMR *NextGenMap* computes a pairwise sequence alignment score. For highest scoring CMRs the corresponding alignment is computed and output.
To reduce computation time without compromising the number of correctly mapped reads we introduced four steps. Each of the steps are in principal not new, but we developed and implemented new ideas that lead to the observed performance.

1. Efficient *k*-mer representation
2. Indexing the reference genomes
3. Identification of CMRs
4. Threshold computation
5. Alignment computation

3.2.1 *k*-mer representation & extraction

In *NextGenMap* *k*-mers only consist of the nucleotides A, C, G, or T; *k*-mers containing other letters according to the IUPAC nucleotide code are excluded. Nucleotides are converted to upper case letters before processing. Soft masking – marking repeats by lower case letters – of reference sequences is ignored. To reduce memory usage and speed up *k*-mer operations like comparison or lookup *NextGenMap* converts nucleotides from the standard 8-bit ASCII to a 2-bit binary representation. The nucleotides A, C, T and G are represented by the binary values 00_2, 01_2, 10_2 and 11_2. Conversion of a single nucleotide can be done efficiently using bit operations:

$$N_2 = (N_{ASCII} >> 1) \wedge 11_2$$

where N_2 is a nucleotide in 2-bit representation, N_{ASCII} is a nucleotide in ASCII representation, $>> 1$ shifts all bits one position to the right and $\wedge$ is the logical AND operator. For example, for the nucleotide A with the binary ASCII code of

01000001_2, shifting all bits by one, 001000000_2, and retaining only the two least significant bits results in the 2-bit representation of 00_2.

To store k-mers *NextGenMap* places the 2-bit representation of the k nucleotides in a 64-bit word starting from the least significant bit. The 4-mer $ATGC$ for example is represented by the binary number 00101101_2. Therefore, each k-mer implicitly has a numerical representation. For $ATGC$ the numerical value is 45. In theory the maximal length of a k-mer in *NextGenMap* is 32 nucleotides.

The 2-bit representation allows us to extract adjacent k-mers from a reference sequence or a read – one of the most important operations during read mapping – efficiently. Instead of extracting all k nucleotides for each k-mer at position i, we reuse the $k-1$ already encodes nucleotides from the k-mer at position $i-1$:

$$K_i = (K_{i-1} << 2) \vee N_i$$

where K_i is the 2-bit encoded k-mer ending at position i and N_i the 2-bit encoded nucleotide at position i.

Furthermore, the 2-bit representation allows us to efficiently compute the reverse complement of a k-mer only using bit operations. Another important operation for finding seed words. Computing complementary nucleotides requires only a single XOR operation with the bit pattern 10_2:

$$A \rightleftharpoons T : 00_2 \oplus 10_2 = 10_2 \oplus 10_2 = 00_2$$

$$C \rightleftharpoons G : 01_2 \oplus 10_2 = 11_2 \oplus 10_2 = 01_2$$

We can extend this to compute the complementary sequence of a full k-mer with $0 < k <= 32$ by using the bit pattern $10101010101010101010101010101010_2$ for the XOR operation.

To reverse the sequence of a k-mer we modified a well know approach to reverse the bit order in a 64-bit word [1]. As shown in figure 3.1 this approach first switches the order of the upper and lower 32-bit block of the 64 bit word. Next, it recursively switches the order of the nested 16, 8, 4, 2 and 1 bit blocks. Since we want to

[1] `http://graphics.stanford.edu/~seander/bithacks.html#BitReverseObvious`

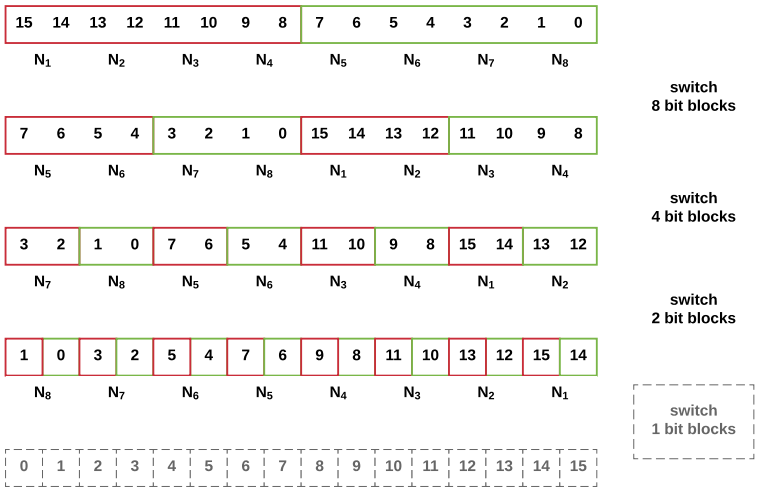


Figure 3.1: Reversing the sequence of a k-mer with bit operations. Illustrated with a 16-bit word containing nucleotides $N_1..N_7$. First, we switch the order of the two 8-bit blocks depicted in green and red. Next, we switch the order of all 4-bit block pairs within the 8-bit blocks. The same is done for all pairs of 2-bit blocks. The last switch (dashed grey lines) is omitted since nucleotides are represented by 2-bits. Each switch requires five bit operations.

reverse the nucleotides encoded by 2-bit, we skip the last step.

3.2.2 Indexing the reference genome

To reduce the size of the hash-table *NextGenMap* by default only considers k-mers that start at every third position ($\Delta = 3$) in the reference genome. Moreover, for indexing we only consider k-mers from the plus strand. To accommodate for reads that map to the minus strand we retrieve the genomic positions of a k-mer and its reverse complement during the CMR search. Our index consists of an array (GP) and a hash-table (HT). GP stores the genomic positions of all k-mers and thus its size equals the number of k-mer positions in the reference genome. Positions where the same k-mer starts are stored consecutively in GP, we call them k-mer blocks. The order of the k-mer blocks is determined by the lexicographic ordering of the k-mers, starting with $AA\ldots A$ and ending with $TT\ldots T$. HT allows the quick retrieval of the genomic positions of a k-mer. A hash function maps the k-mer to the corresponding entry in HT, which contains the index of the entry in GP that

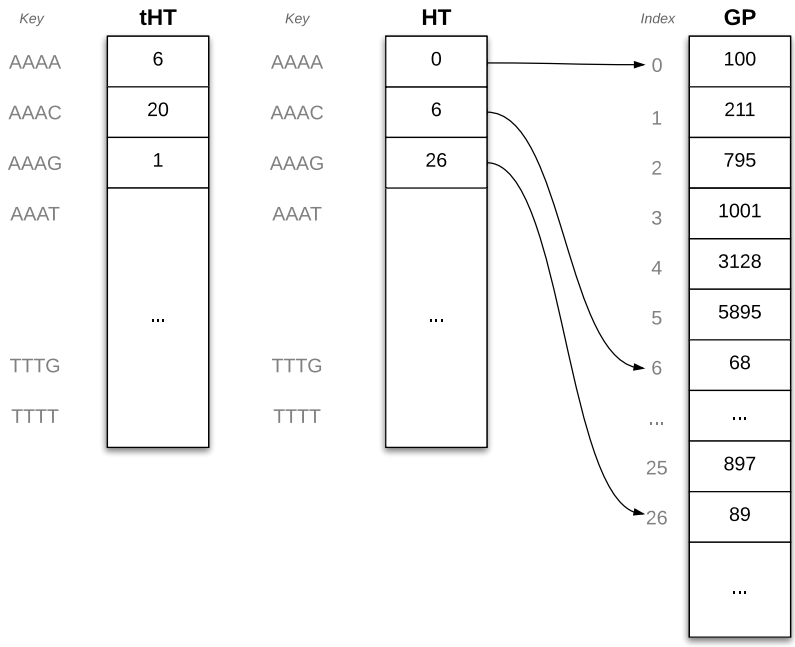


Figure 3.2: Index structure to efficiently compute CMRs.

stores the first genomic position of the k-mer. As an example, assume that *AAAA* occurs six times in the reference genome and the next k-mer *AAAC* occurs 20 times, then HT contains the index 0, identified by key_{AAAA} and the index 6 identified by key_{AAAC}. $GP(0\ldots5)$ stores the genomic positions of *AAAA* and $GP(6\ldots26)$ the positions of *AAAC* (see Figure 3.2).

To build our index, *NextGenMap* first determines the k-mer frequencies. To this end, we introduce a temporary hash-table (tHT) where each entry stores the number of times the corresponding k-mer occurs. Second, *NextGenMap* tags k-mers that occur more than $1,000$ times as repetitive. Repetitive k-mers will be excluded from subsequent steps, because they do not contribute to the identification of the correct mapping positions. Third, *NextGenMap* initializes HT. To this end, *NextGenMap* iterates through tHT. The entry in HT that corresponds to the current entry in tHT is assigned the cumulative number of k-mer positions found in tHT so far not counting the current number.

To continue our example $tHT(AAAA) = 6$, $tHT(AAAC) = 20$, then *NextGenMap* stores the indices 0, 6, and 26 identified by the keys key_{AAAA}, key_{AAAC}, key_{AAAG} in HT. Finally, *NextGenMap* parses the reference genome and stores the genomic position of each k-mer in GP according to the indices in HT. If *AAAA* occurs at positions 100, 211, 795, 1001, 3128 and 5895 then these numbers constitute the first k-mer block in GP (see Figure 3.2).

Our index retrieves the genomic positions for a k-mer with one consecutive memory access to GP. Thus, we optimally exploit the CPU cache.

To save memory HT consists of unsigned 32-bit integer numbers. Therefore, the number of entries in GP is limited to $2^{32} = 4,294,967,296$.

3.2.3 Identification of CMRs

Figure 3.3 illustrates how *NextGenMap* efficiently determines CMRs in a self-contained and intelligible way. More specifically, we map the read R of length $l = 8$ to the reference genome G, assuming step-size $\Delta = 2$ and considering 3bp long k-mers in the reference genome.

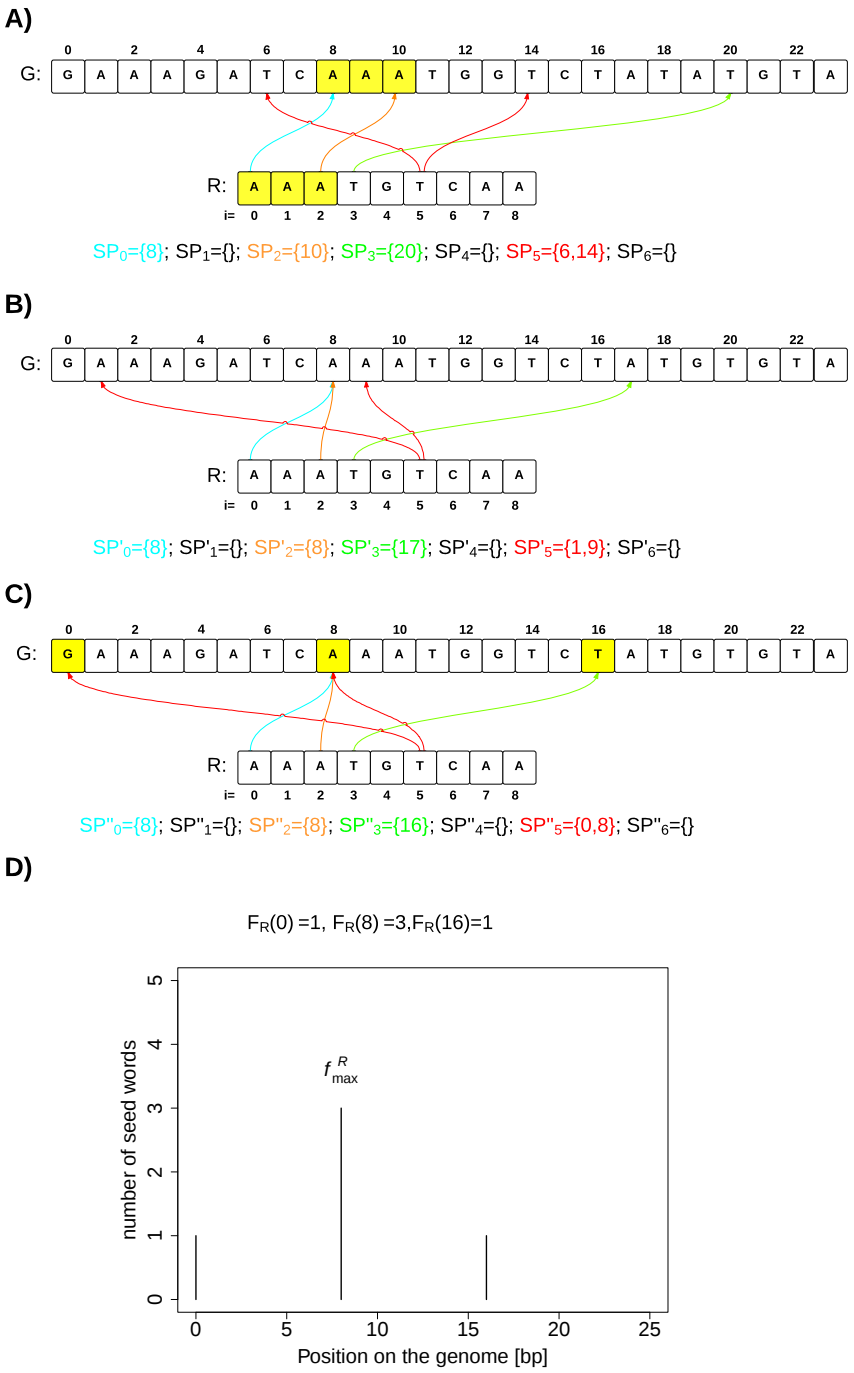


Figure 3.3: Identification of candidate-mapping regions in the reference genome (G) for an individual read (R) using a k-mer size of 3bp and a step size of $\Delta = 2$: (A) Detection of seed words. (B) Shift to the potential read start. (C) Accounting for insertions or deletions. (D) Computing the seed-word distribution F_R along the reference genome G.

The arrows in Figure 3.3(a) shows the start of the seed-words in R and G. Note the 3-mer AAA at position 1 in the reference is not detected due to $\Delta = 2$. The set $\mathscr{SP}_i$ collects the genomic starting positions of the seed-word beginning at position $i = 1,\ldots,(l-k)$ of the read (last line Figure 3.3(a)).

Because we want to align R to G, we compute the potential starting point of the read in the reference genome with respect to the elements in $\mathscr{SP}_i$. That is, we compute the shifted sets $\mathscr{SP}'_i = \{p-i | p \in \mathscr{SP}_i\}$ for each i. The last line in Figure 3.3(b) shows the result. The shift concentrates seed-words at certain positions in the genome, for example $\mathscr{SP}'_0 = \mathscr{SP}'_2$ an indication that R if aligned starting at genomic position 8 will have at least two seed-words.

Finally, to account for insertions or deletions, that may be prevalent for highly polymorphic genomes or sequencing technologies e.g. Ion Torrent, we collect the seed-words of R that occur in the segments $[8j, 8j+1, 8j+2, 8(j+1)-1]$ $(j = 0,1,\ldots)$ at the starting point $8j$ of the segment (highlighted yellow in Figure 3.3(c)). Lumping seed-words modulo 8 leads to an accumulation of seed-words at fewer genomic positions. More formally, we finally compute how often, $F_R(p'')$, the genomic position p'' occurs in all $\mathscr{SP}''_i$'s. Figure 3.3(d) shows the resulting frequency distribution (F_R) for the toy example. We note that the segmentation of the reference genome can be done efficiently using bit-shift operations, that is why we selected the particular value $8 = 2^3$

Computing the seed-word distribution F_R

To compute F_R, we construct a hash-table of length L and process all starting positions. The hash key is computed using a multiplication hashing approach with the parameter suggested by Knuth (1997):

$$key_j = (2,654,435,761 \cdot p'') \bmod L,$$

where $p'' \in \mathscr{SP}''_i$. For each p'' we increase by one the corresponding hash value, which represents $F_R(p'')$. Linear probing is applied to resolve collisions. Collisions occur, when two different p'' result in the same hash key. L is automatically adjusted to efficiently operate on the hash-table. Initially $L = 2^{16}$. If for a particular read an

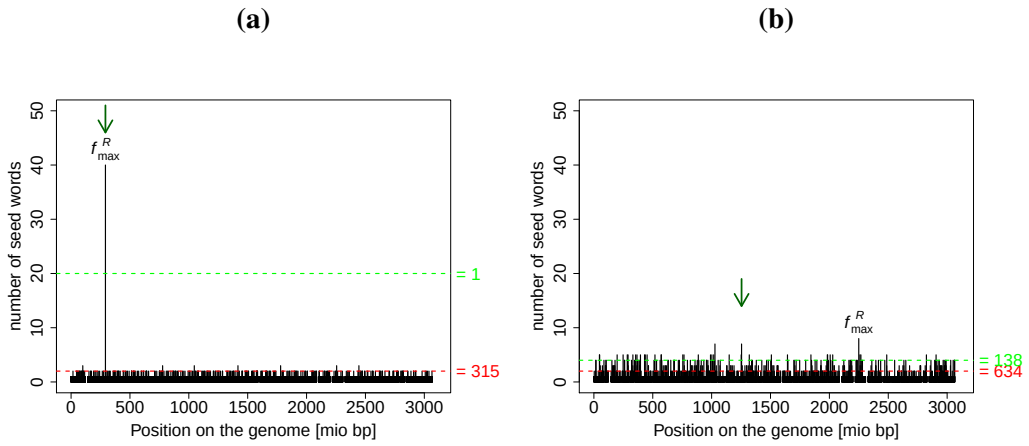


Figure 3.4: Seed-word distribution along the human genome for two reads. The green arrow indicates the correct mapping position. The green line displays the read specific threshold Θ_R. The red line displays a fixed threshold, here 2 seed words. The number of CMRs exceeding the respective threshold are written in green and red.

overflow occurs, we double L for this read and redo the computation. An overflow occurs when $\sum_{i=0}^{l} |\mathscr{S}\mathscr{P}''_i| > L$. If an overflow occurs for 10 out of 100 reads, then the global L is doubled. This reduces the number of re-computations. On the other hand, if no overflow occurs for 100 reads, then L is globally halved. This ensures that the CPU cache is optimally exploited.

Finally, the hash-table lists all genomic starting positions that have at least one seed-word in common with read R. F_R specifies the number of seed-words that support a particular starting position (see Figure 3.3(d)).

3.2.4 Data driven threshold on the number of alignment computations

Figure 3.4 shows the seed-word count distributions F_{R1}, F_{R2} for reads $R1, R2$. The distributions for both reads are quite distinct. F_{R1} shows one clear peek while F_{R2} is uninformative with respect to the potential location of the read. It is intuitively obvious that $R1$ should be aligned to the region around the peak and that it is not necessary to take into account other genomic regions and in fact, the green arrow

(origin of the read) coincides the maximum of F_{R1}. For $R2$ it is not clear, which genomic region is the correct one, thus we need to also compute alignment scores for other genomic regions. We want to optimize the number of alignment computations automatically, taking into account the degree of genomic polymorphism and the read specific characteristics. We note that a firm threshold (red line in Figure 3.4) would cause unnecessarily many alignment score computations for $R1$.
In the following we describe our approach to compute such a read specific threshold Θ_R.
To this end, we notice that if all $k-$mers of a read are pairwise different, the maximal number of seed-word equals

$$S_{\max} = \left\lceil \frac{l-k+1}{\Delta} \right\rceil, \tag{3.1}$$

To elucidate how similar the sequenced reads are to the theoretical $S_{\max}$, we compute for a random sample of B reads $\max\{F_R\}$ for each read R and the average

$$\overline{F_{\max}} = \frac{1}{B} \sum_{j=1}^{B} \max\{F_{R_j}\}, \tag{3.2}$$

where $B = 10,000$.
The ratio

$$\sigma = \frac{\overline{F_{\max}}}{S_{\max}} \tag{3.3}$$

describes how similar the seed-word count of an average read is to the seed-word count of a perfectly matching read ($S_{\max}$). Thus, a small (close to zero) σ indicates that the reads are on average very different from the reference genome, whereas a σ close to one is indicative of reads that are very similar to the reference genome. Large σ values imply that we only need to compute few alignment scores, since we expect situations similar to Figure 3.4a.
However σ describes the average similarity. Figure 3.4 shows reads with very different signals of similarity occur in the same sample. To accommodate for this

observation with the overall similarity σ, we define the read specific threshold

$$\Theta_R = \sigma \max\{F_R\}. \tag{3.4}$$

Finally, *NextGenMap* offers the option to switch off the automatic threshold determination. Then the user can define σ. Setting $\sigma = 1$ implies that only genomic regions with the highest F_R are further processed (fast mode), setting $\sigma = 0$ (slow mode), results in alignment score computations for all genomic positions with $F_R \geq s_{min}$. By default $s_{min} = 1$, but it can also be specified by the user.
To conclude this section: All the genomic positions that exceed the read specific threshold are considered as starting positions for CMRs.

3.2.5 Alignment computation

Finally, the alignment computation identifies the CMR(s) with the highest alignment score and computes the corresponding full alignment(s).
Here, we compute the alignment based on a read and the region on the genome starting from $p'' - c/2$ to $p'' + l + c/2$, where c is the size of the alignment corridor and l is the length of the read. Note that c only restricts the number of consecutive insertion or deletion. The overall number of differences (mismatches, insertion and deletions) are not restricted by c. By default, the alignment corridor is set according to the read length:

$$c = 5 + \bar{l} * 0.15$$

, where $\bar{l}$ is the average read length over all reads. The value can also be modified by the user.
For both, the score and the alignment computation *NextGenMap* uses the MASon library (Rescheneder et al., 2012). MASon utilizes a wide range of CPUs and GPUs independent of the available memory and the maximum number of concurrent computations.
We extended MASon by providing end-to-end alignment algorithms. Furthermore, the user can tune its own preferred scoring parameters. (e.g. for bisulfite treated

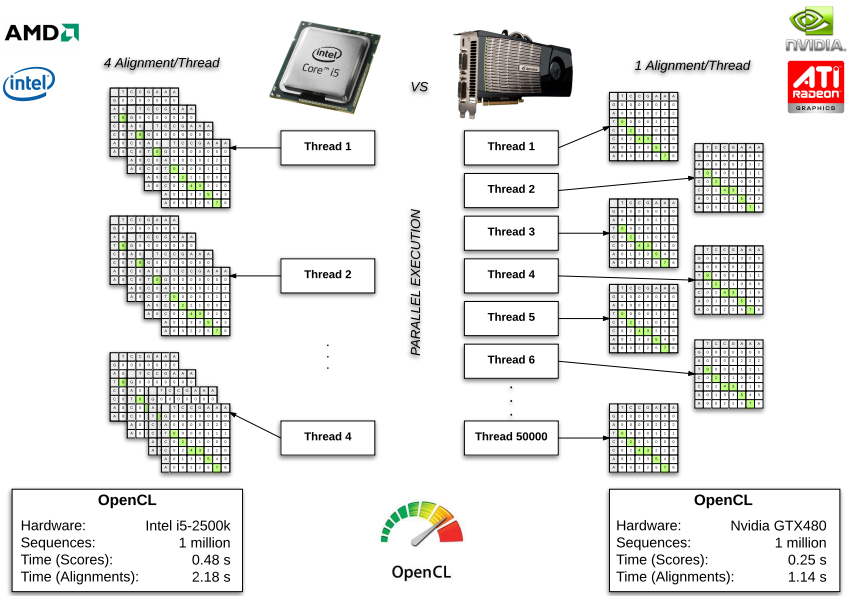


Figure 3.5: Mason efficiently computes short many-to-many alignments. Picture taken from Rescheneder et al., 2012

.

reads the score penalty of T-C or A-G mismatches is reduced).

3.2.6 Mapping quality

NextGenMap computes the mapping quality (MQ) as the relative difference between the alignment score of the best alignment (AS_1) and the score of the second best alignment (AS_2). While mapping quality is defined to be in the range of 0 to 255 most tools use 60 as the maximum. To stick to this convention *NextGenMap* scales the relative difference to a range from 0 to 60:

$$MQ = 60 * \frac{AS_1 - AS_2}{AS_1} \tag{3.5}$$

Thus, if AS_1 and AS_2 are very similar MQ will be low indicating low confidence in the mapping location. If AS_2 is much smaller than AS_1, MQ will approach 60 indicating high confidence. If only a single CMR was identified for a read, meaning AS_2 does not exist, the MQ is set to 60.

3.2.7 Paired-end mapping

When mapping paired-end reads *NextGenMap* first retrieves alignment scores for all CMRs identified for both reads in the pair. Next, *NextGenMap* enumerates all possible pairs of mapping locations found for both reads and discards all pairs with incorrect read orientations and insert sizes that exceed a user-defined threshold. Out of the remaining pairs *NextGenMap* picks the one that has the highest joined mapping score. If multiple pairs have the same joined alignment score, *NextGenMap* chooses the pair with the insert size that is the closest to the average insert size identified in the dataset so far. Finally, *NextGenMap* outputs both alignments and mapping locations as a paired alignment according to the SAM specification. If no proper pair is found *NextGenMap* outputs the top alignment positions for both reads and marks them as a broken pair as defined by the SAM specification.

3.2.8 Benchmark datasets and tools

Data	Genome	Technology	Reads		Source
			number [mio]	length [bp]	
R_1	human	Illumina	29.6	35	SRA
R_2	human	Illumina	10.3	76	SRA
R_3	human	Illumina	10.0	101	SRA
R_4	human	Ion Torrent	0.6	5-396	Life Tech.
R_5	human	454	5.4	51-4,935	SRA
S_1	human	Illumina	20.0	150	Mason
S_2	human	Illumina	20.0	250	Mason
S_3	*Arabidopsis thal.*	Illumina	20.0	100	Mason
S_4	*Drosophila mel.*	Illumina	24.0	100	Mason

Table 3.1: Summary of the data sets used for the benchmarking study.

To study the performance of *NextGenMap* on re-sequencing projects we selected 5 real data sets and 4 simulated data sets. Three Illumina (R_1, R_2, and R_3), one Ion Torrent (R_4), and one 454 dataset (R_5) from the Sequence Read Archive (SRA-NCBI) served as benchmark data. Furthermore, to assess the mapping accuracy we simulated four read sets using Mason (Holtgrewe, 2010), S_1 (150 bp reads), S_2 (250

bp reads) derived form the human genome, S_3 (100 bp reads) from the *A thaliana* genome, and S_4 (100bp) from the *D melanogaster* genome. In all data sets we introduced a 1.2% sequencing error and assumed a genomic polymorphism (SNPs, insertion and deletions) of 0.1% (S_1,S_2) and 2% (S_3, S_4). Finally, we simulated 11 data sets ($A_1,\ldots,A_{11}$) based on the *A thaliana* genome with an increasing degree of genomic polymorphisms (0% - 10%). Total number of reads and read length for all datasets are summarised in Table A.1.

Based on the findings from Nielsen et al. (2011) we compared *NextGenMap* with four popular mapping methods. Representatives of BWT methods were: BWA (version 0.5.9) (Li and Durbin, 2009), its extension for longer reads BWA-SW (version 0.5.9) (Li and Durbin, 2010), and Bowtie2 (version 2.0.0-beta6) (Langmead and Salzberg, 2012). As alignment based representative we selected Stampy (version 1.0.17) (Lunter and Goodson, 2011). We executed all programs with default parameter settings, as suggested by Fonseca et al. (2012). Since Stampy (version 1.0.17) does not support multi-threading we used the "–processpart" parameter to split the dataset and executed four instance in parallel. *NextGenMap* (version 0.4.2) was also executed with its default parameter settings which are: a k-mer size of 13bp and a step-size of $\Delta = 3$. All other parameters are automatically chosen as described. Computations were executed on a desktop computer (AMD Phenom II X4 965, 16 GB RAM, NVidia GTX 460) using 4 threads.

To the best of our knowledge, two other mappers are available that use a graphic card: SARUMAN (Blom et al., 2011) and SOAP3 (Liu et al., 2012). We did not use SARUMAN because it is only designed for mapping reads to microbial genomes and can not efficiently handle eukaryote genomes. In addition only binaries of SARUMAN are available which crashed when we tried mapping our benchmark datasets. SOAP3 was excluded because it was developed for a specific hardware configuration and therefore our computers do not meet the hardware requirements.

3.3 Results & Discussion

In terms of runtimes *NextGenMap* outperformed all analyzed mappers (see Table 3.2) on all real ($R_1,\ldots,R_5$) and simulated data sets ($S_1,\ldots,S_4$ and $A_1,\ldots,A_11$). The

Dataset	BWA	BWA-SW	Bowtie2	Stampy	*NextGenMap* (*CPU only*)	*NextGenMap* (+*GPU*)
R_1	29	35	19	270	12	**10**
R_2	19	35	11	128	8	**7**
R_3	75	48	18	262	13	**11**
R_4	-	104	32	812	28	**18**
R_5	-	117	70	1026	48	**25**
S_1	151	190	61	961	33	**29**
S_2	476	370	107	2,193	65	**53**
S_3	31	67	14	278	8	**5**
S_4	56	81	39	290	17	**8**

Table 3.2: Runtimes in minutes. We highlighted the shortest runtime per data set. We excluded the results from BWA for R_4 and R_5 as it was developed for Illumina reads only.

CPU implementation was 1.1 to 2.3 times faster than Bowtie2, the fasted method so far. The GPU implementation further reduces the runtime (1.6 to 4.9 times faster). If we compare the runtimes of the methods that showed the highest accuracy, *NextGenMap* is between 2.9 and 5.8 times faster than BWA-SW. Stampy, though very accurate for high polymorphic genomes (2.0%), shows the longest runtimes (maximum 37 hours) compared to *NextGenMap* (65 minutes for the same data set).

Table 3.3 displays the mapping accuracies for the benchmark study. For a low genomic polymorphism (S_1 and S_2) BWA-SW shows shows 0.1% and 0.2% more correctly mapped reads compared to *NextGenMap* for S_1 and S_2 respectively. However, for genomic polymorphism of 2% the alignment based mappers (*NextGenMap*, Stampy) are the best with a mapping accuracy of 97.6% and 85.5% for S_3 and S_4 respectively.

To further elucidate the influences of polymorphic genomes on the mapping accuracy, we varied the degree of genomic polymorphism from 0% to 10% for *A. thaliana* ($A_1,\ldots,A_{11}$). Figure 3.6a display the decline in mapping accuracy for BWT based methods and also shows that accuracy is unaffected by degree of genomic polymorphism for the alignment based mappers (see Table A.2 and Table A.3 for exact numbers). However, Stampy can only retain the accuracy with the expense of an increased computing time (Figure 3.6b). On the contrary, *NextGenMap* shows no substantial increase in runtime. In summary, *NextGenMap* maps reads very accurately and independent of the amount of genomic polymorphism (up to

Program:		**BWA**	**BWA-SW**	**Bowtie2**	**Stampy**	***NextGenMap*** *(CPU only)*	***NextGenMap*** *(+GPU)*
S_1	correct map (%)	83.6	**98.3**	97.6	97.8	98.1	98.1
	¬ map (%)	15.0	0.0	0.3	0.1	0	0
	wrong map (%)	**1.4**	1.7	2.1	2.1	1.9	1.9
S_2	correct map (%)	69.2	**99.0**	98.7	98.9	98.9	98.9
	¬ map (%)	30.1	0.0	0.0	0.0	0.0	0.0
	wrong map (%)	**0.7**	1.0	1.3	1.1	1.1	1.1
S_3	correct map (%)	58.0	93.7	94.7	**97.6**	**97.6**	**97.6**
	¬ map (%)	40.6	3.8	2.9	0.0	0.0	0.0
	wrong map (%)	**1.4**	2.5	2.4	2.4	2.4	2.4
S_4	correct map (%)	50.9	81.9	82.6	**85.5**	**85.5**	**85.5**
	¬ map (%)	40.3	3.5	3.0	0.6	0.0	0.0
	wrong map (%)	**8.8**	14.6	14.4	13.9	14.5	14.5
R_1	map (%)	90.8	81.8	92.7	88.9	**97.4**	**97.4**
	¬ map (%)	9.2	18.2	7.3	11.1	2.6	2.6
R_2	map(%)	93.2	95.7	96.8	94.6	**97.4**	**97.4**
	¬ map (%)	6.8	4.3	3.2	5.4	2.6	2.6
R_3	map(%)	69.4	91.6	90.9	85.7	**96.6**	**96.6**
	¬ map (%)	30.6	8.4	9.1	14.3	3.4	3.4
R_4	map(%)	-	96.1	98.3	97.4	**99.9**	**99.9**
	¬ map (%)	-	3.9	1.7	2.6	0.1	0.1
R_5	map(%)	-	97.8	99.0	90.0	**99.8**	**99.8**
	¬ map (%)	-	2.2	1.0	10.0	0	0

Table 3.3: Percent of correctly mapped, unmapped (¬ map) and wrongly mapped reads for $S_1,\ldots,S_4$ and percent of mapped and unmaped reads for $R_1,\ldots,R_5$.

10%). At the same time *NextGenMap* exhibits the fastest runtime of all tested tools. We also note that graphic cards provide additional speedup. However, a strong effect is only observed when the number of alignment computations is high, as is the case for highly polymorphic genomes (8% or more).

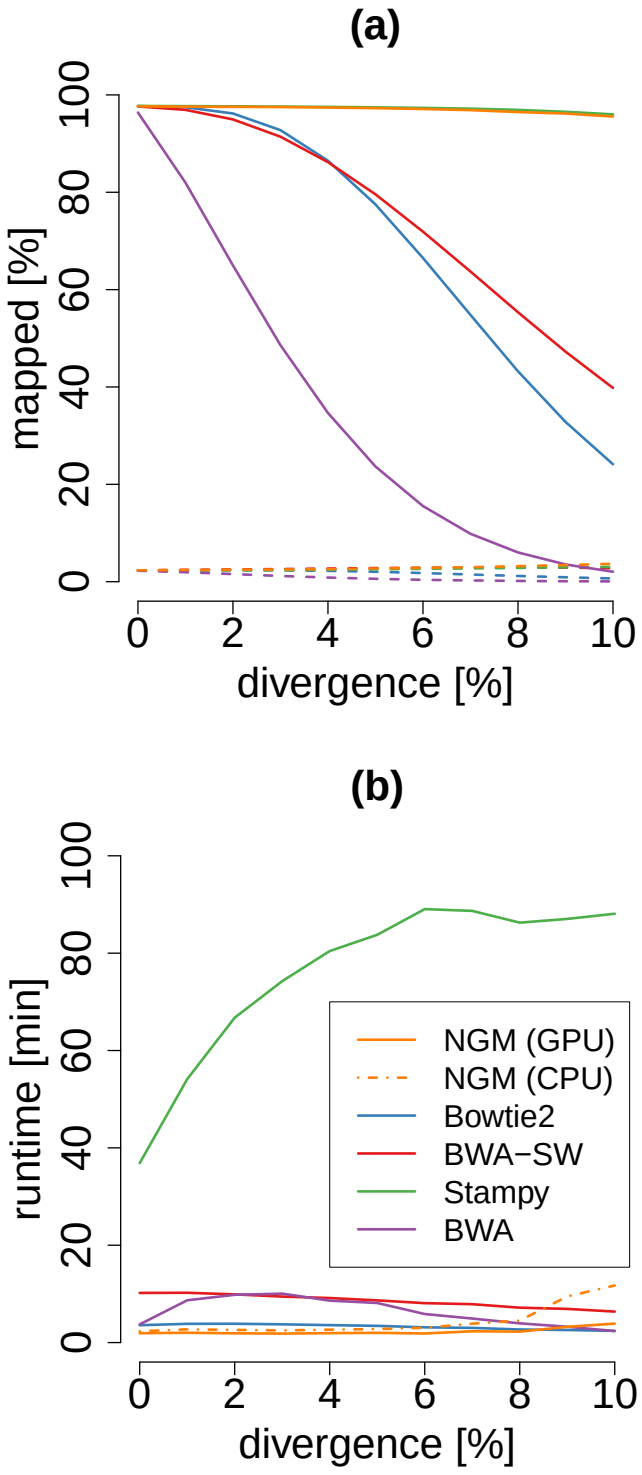


Figure 3.6: (a) Percent of correctly (solid) and incorrectly (dashed) mapped reads and (b) running times for different degree of genomic polymorphisms between read and reference genome for five million 100bp *A. thaliana* reads ($A_1, \ldots, A_{11}$).

3.4 Summary

Here we showed with real and simulated data that *NextGenMap* maps High Throughput Sequencing reads very accurate and fast at the same time. Finally, the automatized adjustment of the parameters (e.g. minimal *k*-mer matches, alignment corridor) based on the input and the automatic adaptation to the hardware results in a reliable and fast read mapper with minimal user interaction. This allows *NextGenMap* to map reads reliably even to highly polymorphic genomes (10%). Thus it may also be used to map reads from non-standard organism to a phylogenetically close genome or to apply it to metagenomics data.

Chapter 4

Accurate detection of complex structural variations using single-molecule sequencing

4.1 Introduction

Structural variations (SVs), including insertions, deletions, duplications, inversions, and translocations at least 50 bp in size, account for the greatest number of divergent base pairs across human genomes (Weischenfeldt et al., 2013). SVs contribute to polymorphic variation; pathogenic conditions; large-scale chromosome evolution (Lupski, 2015); and human diseases such as cancer (Macintyre et al., 2016), autism (Hedges et al., 2012), and Alzheimer's (Rovelet-Lecrux et al., 2006). SVs also affect phenotypes in many other organisms (Sudmant, 2015, Dennenmoser, 2017, Jeffares et al., 2017, Zichner et al., 2013, Imprialou et al., 2017).

In one of the first reports of SV prevalence, published in 2004, Sebat et al. discovered in a microarray study that large-scale copy-number polymorphisms are common across healthy human genomes. Today, SV detection most often uses short paired-end reads. Copy-number variations are observed as decreases (deletions) or increases (amplifications) in aligned read coverage (Kadalayil et al., 2015), and other types of SVs are identified by the arrangement of paired-end reads or split-

read alignments (Alkan et al., 2011, Layer et al., 2014, Rausch et al., 2012, Chen et al., 2016). Short read SV callers, however, have been reported to lack sensitivity (only 10% (Huddleston, 2017) to 70% (Sudmant, 2015, Jeffares et al., 2017) of SVs detected), exhibit very high false positive rates (up to 89%) (Sudmant, 2015, English et al., 2014, Mills et al., 2011, Tattini et al., 2015, Teo et al., 2012), and are unable to capture complex or nested SVs (Sudmant, 2015, Lucas Lledó and Cáceres, 2013).

Long-read single-molecule sequencing has the potential to substantially increase the reliability and resolution of SV detection. With average read lengths of 10 kbp or higher, the reads can be more confidently aligned to repetitive sequences which often mediate the formation of SVs (Lucas Lledó and Cáceres, 2013). Long reads are also more likely to span SV breakpoints with high-confidence alignments. Despite these advantages, long reads introduce new challenges. Most importantly, they have a high sequencing error rate - currently 10-15% for Pacific Biosciences (PacBio) and 5-20% for Oxford Nanopore sequencing (Goodwin et al., 2016) - which necessitates new computational tools. A number of long-reads aligners are available including LAST (Kiełbasa et al., 2011), BlasR (Chaisson and Tesler, 2012), BWA-MEM (Li, 2013), GraphMap (Sović et al., 2016) and MECAT (Xiao, 2017). However, none of them have been optimised for SV calling and thus fail to accurately align reads spanning SVs. Furthermore, only one stand-alone SV calling tool, PBHoney (English et al., 2014), is available to detect all types of SVs from long-read data. Other tools such as SMRT-SV (Chaisson, 2015) have been proposed but only support a subset of all SV types.

To address these challenges, we introduce two open-source algorithms, NGMLR and Sniffles, for comprehensive long-read alignment and SV detection. NGMLR is a fast and accurate aligner for long-reads based on extension of our previous short-read aligner, NextGenMap (Sedlazeck et al., 2013), with a new convex gap-cost scoring model to align long reads across SV breakpoints. Sniffles successively scans the high accuracy alignments to identify all types of SVs. Its SV-scoring scheme evaluates candidate SVs on the basis of their size, position, type, coverage, and breakpoint consistency, and thus overcomes the high insertion/deletion (indel) error rates in long-read sequencing.

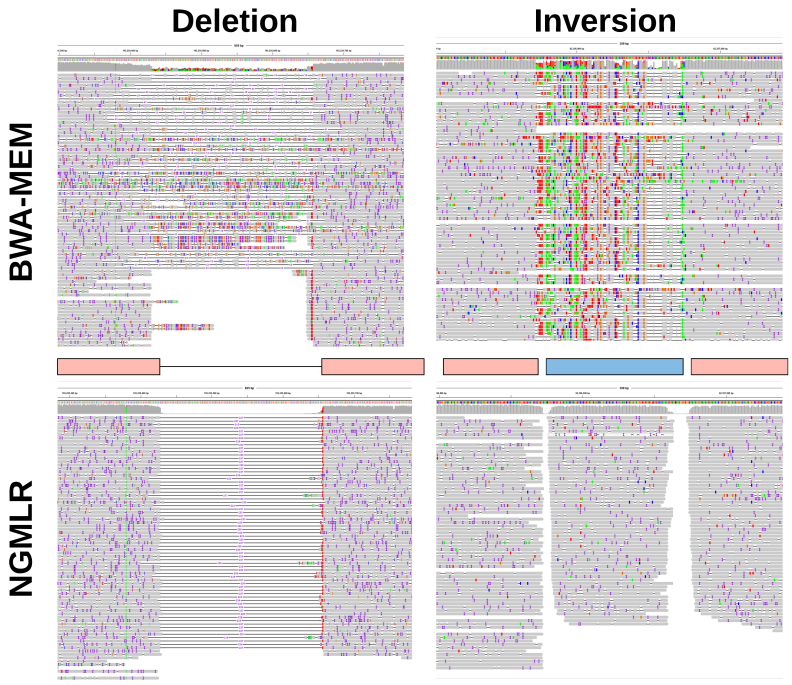


Figure 4.1: Example regions showing alignments spanning a 228-bp deletion (left) and a 150-bp inversion (right). BWA-MEM alignments (upper tracks) show alignment artefacts in reads spanning the two SVs. NGMLR alignments are free of artefacts and thus enable accurate calling of the two SVs.

We applied NGMLR and Sniffles to simulated and genuine datasets for *Arabidopsis*, healthy human genomes, and a cancerous human genome to demonstrate their increased accuracy compared to alternate short- and long-read callers. A particularly innovative feature of Sniffles is its ability to detect nested SVs, such as inverted tandem duplications (INVDUPs) and inversions flanked by indels (INVDELs). These are poorly studied classes of SVs. Although both have been previously associated with genomic disorders (Carvalho et al., 2011, Shimojima et al., 2012, Carvalho and Lupski, 2016, Mühle et al., 2007), they could not be routinely detected, and so their full significance is currently unknown. Finally, we show that our methods reduce the sequencing and computational costs required per sample, and thus make the application of long reads to large numbers of samples increasingly feasible.

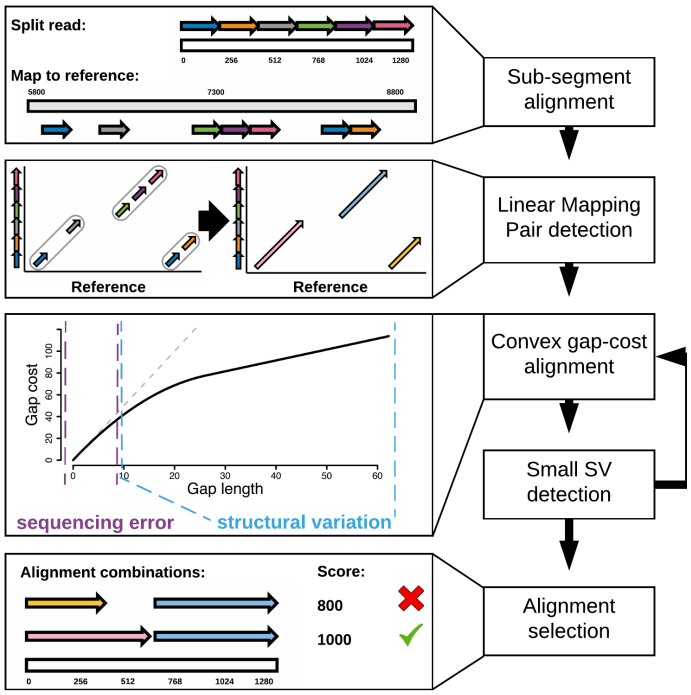


Figure 4.2: Overview of the read mapping algorithm implemented by NGMLR.

4.2 Methods

4.2.1 Accurate alignment of noisy long reads

NGMLR uses a convex gap-scoring model (Gusfield, 1997) to accurately align reads spanning genuine structural variation in the presence of small indels (1-10 bp) that commonly occur as sequencing errors (Figure 4.1 bottom left). Larger or more complex SVs are captured through split-read alignments (Figure 4.1 bottom right).

Similar to other tools (Chaisson and Tesler, 2012), to achieve both high performance and accuracy, NGMLR first partitions long reads into short subsegments and aligns them independently to the reference genome (Figure 4.2). It groups colinear subsegment alignments into longer segments, which are then aligned using dynamic programming and our convex gap-cost scoring scheme. Finally, NGMLR selects the highest-scoring non-overlapping combination of aligned segments per read and

outputs the results in standard SAM/BAM format. In the following, we describe the details of alignment computation in NGMLR.

Detection of linear mapping pairs

Sub-segment alignment: To identify local similarities between a long read and the reference genome, NGMLR splits each read into non-overlapping 256 bp sub-segments and maps them to the reference genome independently of each other using the seed and vote approach described by Sedlazeck et al. (2013). Figure 4.3a shows a toy example for a read of length 1,536 that was split into six sub-segments. Briefly, a sub-segment is decomposed into all overlapping *k*-mers (13-mers per default). For each *k*-mer, the location(s) for that *k*-mer on the reference genome are retrieved from a hash table index data structure. All regions of the reference genome that exceed a certain number of *k*-mer matches are considered candidate mapping regions (CMR) for the sub-segment. Next, a pairwise local alignment score for all CMRs and the sequence of the sub-segment is computed. If the alignment score does not exceed a minimal threshold the CMR is discarded. NGMLR sorts all remaining CMRs based on their alignment score and retrieves the highest score. All CMRs with a score lower than 75% of the highest score found for their respective sub-segment are discarded. We call all remaining CMRs "anchors" between the sub-segment and the reference genome. An anchor is described by its starting position on the long read, its mapping position on the reference genome, its mapping orientation, and its alignment score (Figure 4.3a bottom). Sub-segments that map to highly repetitive regions with more than 1000 (default) anchors are discarded, as they are not informative for finding local similarities between the read and the reference. Note, short stretches of higher error rates sometimes observed in long reads could prevent anchor detection. As this will be relevant for accurate alignments later, Figure 4.3a shows an example of a sub-segment (orange arrow) where NGMLR was not able to identify any anchors.

Building linear mappings pairs: Next, NGMLR identifies all segments of the read that are not interrupted by a structural variation and can therefore be repre-

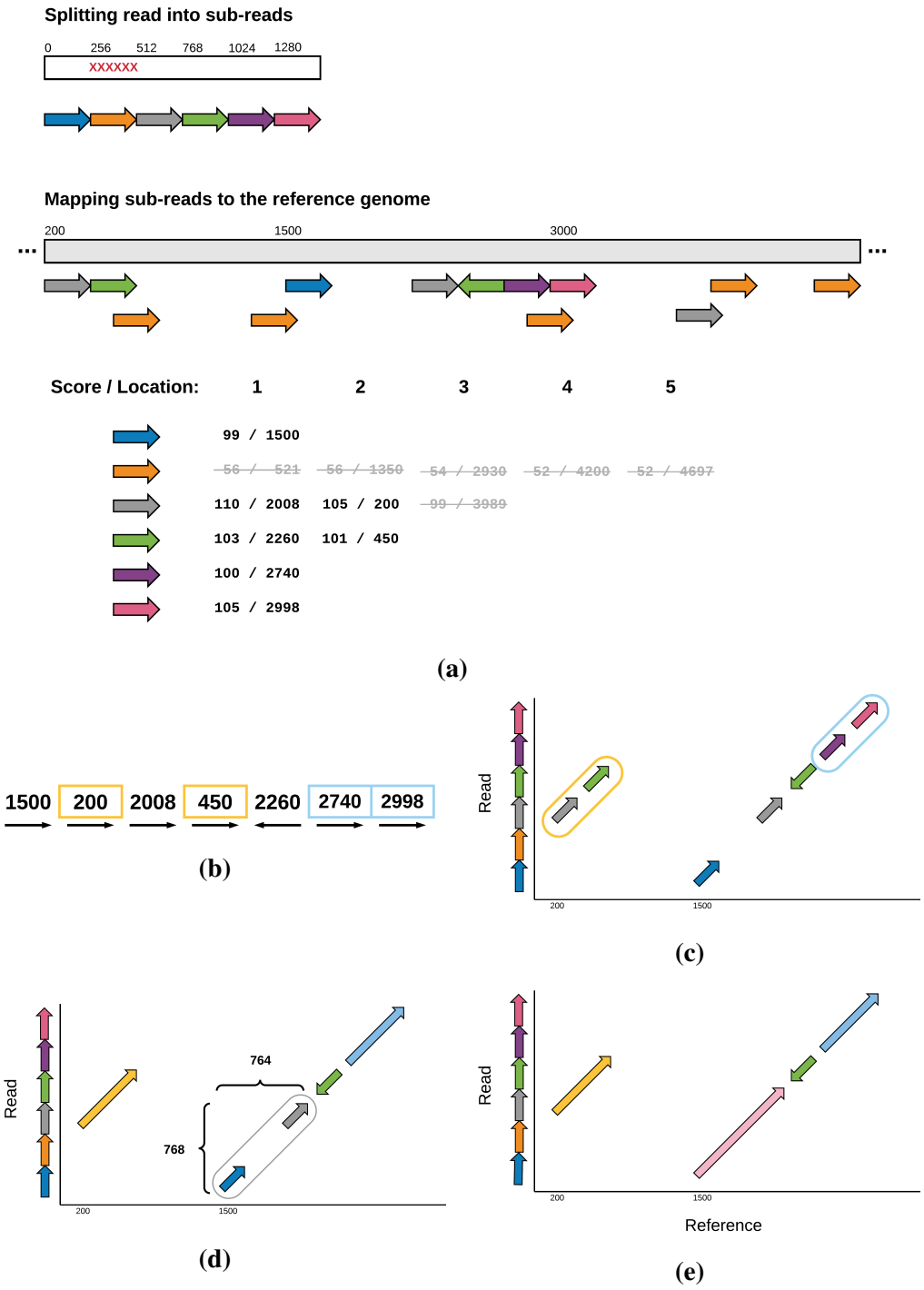


Figure 4.3: NGMLR workflow for detecting linear mapping pairs (LMPs). (a) Reads are split into sub-segments and aligned to the reference genome (a). A modified longest increasing subsequence algorithm detects sub-segments that map co-linearly to the reference sequence to detect LMPs (b and c). LMPs located on the same diagonal in the alignment matrix are merged (d) to form the final set of LMPs (e).

sented by a single linear alignment to the reference genome. We call these pairs of segments of the read and regions on the reference linear mapping pairs (LMP). To identify LMPs, NGMLR identifies the largest set of anchors that map co-linearly to the reference genome. In other words, NGMLR looks for high quality sub-segment mappings that are located on the same diagonal in a hypothetical dot plot of the read and the reference genome. The search for anchors that map in the same order to the read and the reference can be implemented by sorting the anchors based on their position on the read and searching for the longest increasing subsequence (LIS) (Gusfield, 1997) of their respective reference coordinates (Figure 4.3b). To enforce co-linearity between anchor mappings, we extended the basic LIS algorithm to include the following restrictions: (1) Two subsequent anchors can only be included in the LIS if they are on the same strand and (2) if the distance between their starting positions on the read and distance between their mapping location on the reference genome deviates by only 25% of the sub-segment length. This ensures that the two anchors are not separated by a structural variation. To avoid merging of two unrelated anchors, we further require their distance on the reference genome and on the read, be less than two times the length of the sub-segment. This constrained longest increasing subsequence algorithm allows us to identify the largest set of co-linear anchors mappings. Joining this set gives us the longest LMP of the read. As a read that spans a structural variation might generate more than one LMP, NGMLR removes all anchors that have been used to form a LMP and repeats the above step until it is unable to find any more LMPs with support from at least two anchors. Figure 4.3c shows a visualisation of this process as a dot plot between the read and the reference. Anchors are shown as arrows. Coloured circles show the two LMPs detected in our toy example.

Merging compatible linear mapping pairs: So far, NGMLR has identified a set of LMPs that do not span structural variations and are therefore guaranteed to align linearly to the reference genome. However, for a sufficiently long read, insertions and deletions shorter than the read length can be part of a linear local alignment. Spanning these shorter SVs with a linear alignment is preferable to creating two split alignments as it makes SV detection easier for downstream tools. Further-

more, missed anchors (orange arrow in Figure 4.3a) will artificially split LMPs. To identify the minimal number of linear local alignments needed to correctly map a read NGMLR next looks for pairs of LMPs that are separated by short indels or were falsely split because of sequencing error and merges them. To this end, NGMLR clusters all LMPs into subsets such that within a subset all LMPs are in a, per default, 8000 bp wide corridor in our hypothetical dot-plot between the read and the reference sequence. Next, NGMLR sorts the LMPs within each subset by their start location on the read and iteratively attempts to merge adjacent LMPs. The relative location of two adjacent LMPs can indicate whether they are separated by a structural variation or not. If the distance between two LMPs on the read and on the reference is the same, meaning they align collinearly, this indicates that they are not separated by a structural variation (Figure 4.3d dark blue and grey anchors). Thus, NGMLR will join them. In contrast, a larger distance on the reference indicates a deletion, while a larger distance on the read indicates an insertion. In this case, NGMLR joins two LMPs only if the size of the insertion or deletion is smaller than both LMPs. LMPs on opposite strands are never joined (Figure 4.3c grey, green and light blue anchors).

Extending linear mapping pairs: LMPs frequently do not contain the first and the last few base pairs of a long high error rate read. Therefore, NGMLR extends all LMPs by two times the sub-segment length. Furthermore, NGMLR closes all gaps between two adjacent LMPs that are within an alignment corridor but were not merged (e.g., because of different strands) to form the final set of LMPs.

Computing pairwise alignments with convex gap costs for LMPs

Alignment using a convex gap-cost model: For each LMP we know its approximate start and end position on the read and its approximate start and end position on the reference genome but not the specific sequence alignment. Therefore, in the next step, for each LMP, NGMLR extracts the read sequence and the reference sequence and uses a Smith-Waterman-like dynamic programming algorithm to compute their pairwise sequence alignment. When aligning long-reads it is crucial

to choose an appropriate gap model as there are two distinct sources of insertions and deletions (indels): Sequencing error predominantly causes very short randomly distributed indels (1-5bp), while biological structural variations cause longer indels (20bp+). Furthermore, for indels caused by structural variations it is more likely to find one large indel than two smaller indels in close proximity.

Currently, two gap models are mainly used: linear and affine gap cost models. A linear gap cost model - where the cost of a gap with length L equals the cost of L gaps with length 1 - appropriately models indels originating from sequencing error. However, linear gap costs favour shorter gaps and therefore cause long indels stemming from SV to be falsely split into several smaller indels. Affine gap costs more realistically model indels from SVs by introducing a separate penalty for gap opening and gap extension. However, for long-reads the effect of the higher gap-open penalty is outweighed by the cost of the gaps from the many sequencing errors, causing them to be falsely clustered. Therefore, the affine gap cost only has a small effect on longer indels, especially when indels are located in regions of low sequence complexity. Figure 4.4a shows two different pairwise alignments of the same sequences. Alignment 1 is the correct alignment showing one long deletion stemming from a SV and six 1bp indels stemming from sequencing error. In Alignment 2 the deletion is split into three mid-sized deletions and only four 1bp indels are reported. However, the number of gap openings and gap extensions is the same between both alignments. Therefore, with an affine gap cost model the alignment score of both alignments is the same.

To account for sequencing error and real SVs at the same time, NGMLR uses convex gap-costs for aligning long-reads. Appropriately parameterized, convex gap costs mimic linear gap costs for short indels (e.g., sequencing errors) while at the same time favouring longer gaps for indels stemming from structural variations (Figure 4.4b).

We define the cost of a gap G of length i to be:

$$G(i) = \begin{cases} g_0 & i = 0 \\ G(i-1) + \min \begin{cases} g_M & \\ g_E + g_D * (i-1) & \end{cases} & i > 0 \end{cases}$$

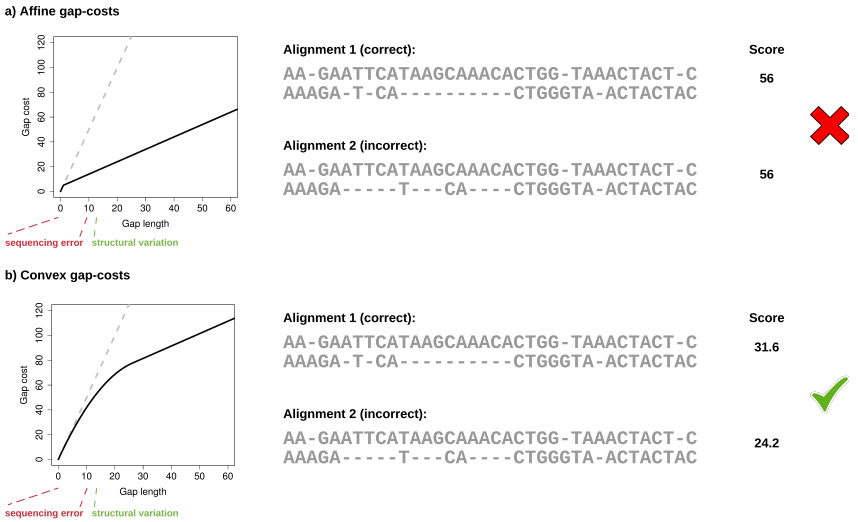


Figure 4.4: Two different alignments for the same sequences with affine gap-costs (a) and convex gap-costs (b). Only with convex gap-costs, the correct alignment shows a higher score than the incorrect alignment.

Similar to the affine gap cost model, g_O only applies to the first gap character while g_E is used for any additional gap character. Note, while we use different names here (g_O and g_E) to make the differences to affine gap costs clear. For all practical purposes we use $g_E = g_O$. In addition, we introduce a gap decay parameter g_D (default: 0.15) that reduces the cost of adding an additional gap character depending on the total length of the gap. In other words, the longer a gap is the lower the penalty of extending it. To prevent the penalty for extending a gap from going to zero, we also introduce the gap min g_M (default: -1) parameter.

For very short insertions and deletions, $|g_D * (i-1)|$ is close to zero. Thus, $g_E + g_D * (i-1) \approx g_O$, meaning our gap model behaves similarly to linear gap penalties (Figure 4.4b, gap length $0-10$). As a result, two gaps of length one are assigned a very similar score as a single gap of length two ($G_1 + G_1 \approx G_2$). This correctly models indels originating from random sequencing error. For medium length gaps and longer insertions or deletion, where $g_E + g_D * (i-1) \approx g_M$, our gap model favors one longer gap over two smaller ones as extending a longer gap becomes much cheaper than opening a new gap ($G_{25} + G_{25} \gg G_{50}$). Therefore, the alignment score of the correct Alignment 1 from Figure 4.4b is higher than the score of the

incorrect Alignment 2.

Available algorithms capable of using this gap model for computing alignments have to scan the full row i and column j of the alignment matrix V to compute the correct score of any cell $V_{i,j}$. This increases the runtime complexity from $O(m^2)$ to $O(m^3)$ for a naive implementation. Using such an algorithm for aligning long-reads to a reference genome is computationally infeasible. Gusfield (1997) describes an improved implementation with $O(m^2 * \log(m))$. Although more favorable in terms on runtime complexity, this algorithm is complex and hard to optimize and therefore still not fast enough for mapping large datasets in practice. Therefore, we adapted a heuristic implementation of the convex gap cost algorithm found in swalign (https://github.com/mbreese/swalign). We follow the approach of linear gap costs were the value of $V_{i,j}$ only depends on $V_{i-1,j}$, $V_{i,j-1}$ and $V_{i-1,j-1}$. However, we define a function $g(l)$ the gives us the penalty of extending an indel of length l by one as follows:

$$g(l) = \begin{cases} g_O & l = 0 \\ \min \begin{cases} g_M & \\ g_E + g_D * l & \end{cases} & l > 0 \end{cases}$$

Furthermore, we introduce two additional matrices to keep track of our length estimates for insertions $I_{i,j}$ and deletions $D_{i,j}$ while computing $V_{i,j}$:

$$D_{i,j} = \begin{cases} D_{i-1,j} + 1, & V_{i,j} = V_{i-1,j} + g(D_{i-1,j}) \\ 0, & otherwise \end{cases}$$

$$I_{i,j} = \begin{cases} I_{i,j-1} + 1, & V_{i,j} = V_{i,j-1} + g(I_{i,j-1}) \\ 0, & otherwise \end{cases}$$

In other words, if the maximum score in $V_{i,j}$ was derived from $Vi-1,j$ we assume that the final alignment will contain a deletion at this position. Therefore, we increase the estimated length of the deletion at cell $V_{i,j}$ by one. Finally, we compute the alignment matrix as follows:

$$V_{i,j} = \max \begin{cases} V_{i-1,j-1} + s(Q_i, S_j) \\ V_{i-1,j} + g(D_{i-1,j}) \\ V_{i,j-1} + g(I_{i,j-1}) \end{cases}$$

where $s(Q_i, S_j)$ is the match score or the mismatch penalty for the reference base i and the read base j. After computing V we search for the element with the highest score and use backtracking to find the full sequence alignment.

Speeding up alignment computation: Computing the full alignment matrix V is infeasible for long read alignments (Chaisson and Tesler, 2012). Since NGMLR knows from the sub-segment alignments that the read and the reference sequence are highly similar, the optimal alignment will be close to the diagonal of the full alignment matrix. Therefore, NGMLR computes a banded alignment centered between the start and end positions of the LMP and extends its width until all sub-segment mappings the LMP is based on are contained. In cases where NGMLR underestimated the bandwidth, the computed alignment will not span the full read sequence but stop at or very close to the border of the alignment band. NGMLR detects such cases during backtracking and then recomputes the alignment with a larger band.

NGMLR further optimizes the computation of the convex alignments by applying vectorization using SSE2 instructions. SSE are a form of SIMD (single instruction, multiple data) and save computational time by processing multiple values simultaneously. The use of SSE allows NGMLR to raise the number of reads that are mapped within the same amount of time, independently of the number of CPU threads used. Computing the alignment matrix is the bottleneck when computing convex gap cost alignments and therefore was chosen as primary target for optimization with SSE. In this step, the alignment matrix is filled with the optimal sub-alignment scores for each letter pair between the read and reference sequence. NGMLR applies vectorization to optimize the forward step, by concurrently computing the values of multiple cells in the alignment matrix. Most of the computationally expensive branching operations, e.g., the selection of the optimal scores, are

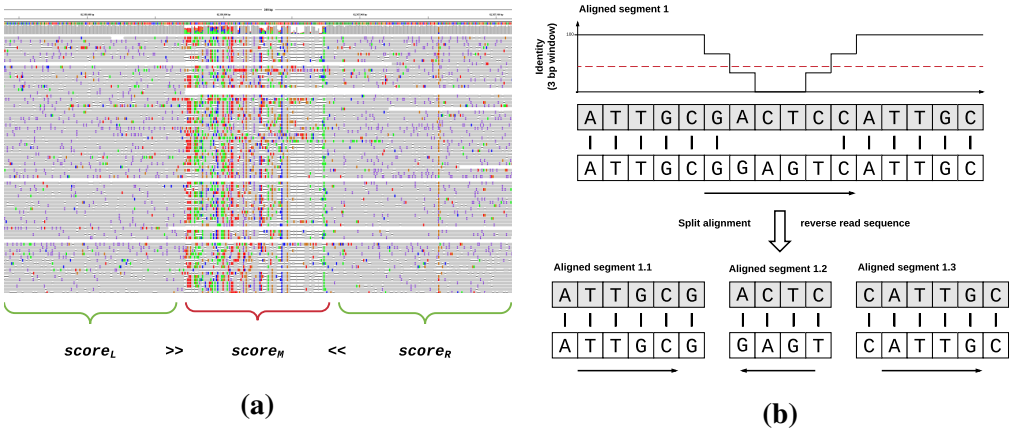


Figure 4.5: Detection of a 5bp inversion in a 17 bp alignment.

further replaced with linear arithmetic operations. The concurrent step is followed up by a regular non-SIMD pass which then resolves the dependencies between the values of the cells which were computed in parallel.

Small inversion detection

The LMP identification and the alignment step account for most types of structural variations. However, short inversions and balanced translocations are difficult to detect as they often do not get sufficient sub-segment support to be identified during the LMP identification step. Therefore, two LMP that are separated by a small inversion or balanced translocations are sometimes falsely merged. Even during the subsequent alignment computation of the merged LMP these SVs stay undetected, especially in long reads, since the score penalty of misaligning e.g., a short inverted segment is small compared to the overall alignment score of the rest of the read (Figure 4.5a). Thus, the segment of the read covering the inversion is forced to align to its reverse complement in the reference genome. This leads to random alignments for the inverted segment and makes it impossible to detect the inversion in down-stream analysis.

The correct way to handle such an inversion is splitting the reads at the borders of the inversion and mapping it in three separate segments. To this end, NGMLR

scans all aligned LMP for regions with a sequence identity smaller 65% (as expected from random alignments). If such a region is detected, NGMLR extracts the respective sequence, computes the reverse complement, and realigns it to the reference genome. If the score increases compared to the original alignment and is above an empirically determined threshold, NGMLR reports the inversion.
In more detail, NGMLR computes for each match, or mismatch position of the alignment its local sequence identity I_i as follows:

$$l_i = \frac{m}{32}$$

Where m is the number of matches found between $i-16$ and $i+16$ in the alignment. Next, NGMLR scans for clusters of positions I with an identity $I_i < 65\%$. Briefly, NGMLR looks for neighboring positions with $I_I < 65\%$ which are separated by not more than 20 bp. If it finds a cluster C that covers at least 40 bp, NGMLR extracts the covered read sequence and aligns this sequence and its reverse complement to the reference sequence. If the alignment score is higher for the reverse complement, the read covers an inversion. Therefore, NGMLR splits the LMP at the borders of the inversion and recomputes the alignments for the three resulting LMPs. Figure 4.5b shows an example of an aligned LMP. The read and the reference sequences are identical except for 4 bp inversion. For simplicity, we chose a 3 bp window to compute the local sequence identity for this example. For the positions of the alignment that cover the inversion the local sequence identity drops (Figure 4.5b top). All positions below the 65 % threshold are extracted. The extracted sequence and its reverse complement are aligned to the reference genome. Since the alignment of the reversed sequence is higher, the LMP is split, and the respective alignments computed (Figure 4.5b bottom).

Selection of linear alignments & Mapping Quality computation

The final set of aligned LMPs contains all linear alignments required for the correct read mapping. However, due to repeats in the reference genome, segments of a read can map to more than one location. For such a segment, NGMLR would detect two independent linear alignments. However, in the final output we want every

nucleotide of the read aligned to exactly one nucleotide in the reference sequence. Therefore, NGMLR must choose the best combination of linear alignments that do not overlap on read coordinates. NGMLR uses a dynamic programming algorithm that determines the non-overlapping set of linear alignments with the maximal joint score by computing the best joint score $S(i)$ for all prefixes of the read that end in position i. For illustration, assume a read R has a linear alignment with score s that starts on the read at position 3000 and ends at position 8000. If we know all $S(j)$ with $j < 8000$:

$$S(8000) = \max \begin{cases} S(7999) \\ S(2999) + s \end{cases}$$

Similarly, $S(1)$ only depends on $S(0)$ and the scores of all linear alignments that end in position 1. Since we know the alignment scores for all linear alignments, and $S(0) = 0$, we can compute $S(1)$ and subsequently $S(i)$ for all prefixes of R. Finally, we use a simple backtracking procedure, to determine the set of linear alignments S was computed from. For each linear alignment of this set, NGMLR finally computes a mapping quality value individually. We define the mapping quality of a linear alignment to be the average mapping quality of all its overlapping sub-segments. Note that this potentially underestimates mapping quality, as the mapping of an LMP can be unique with respect to the genome even if all its sub-segments have a low mapping quality.

4.2.2 Detecting structural variation from long read alignments

Since we focus on read mapping in this thesis, we will give only a brief overview of Sniffles and how it implements SV calling from accurate long read alignments. For more info, please see Appendix B.1. Briefly, Sniffles detects indels, duplications, inversions, translocations, and nested events and can be used with any aligner, although it performs best with NGMLR, as it produces the most accurate alignments. The principal steps consist of scanning the alignments of each read independently for potential SVs and then clustering the candidate SVs across all reads (Figure 4.6). Sniffles uses both within-alignment and split-read information to detect SVs,

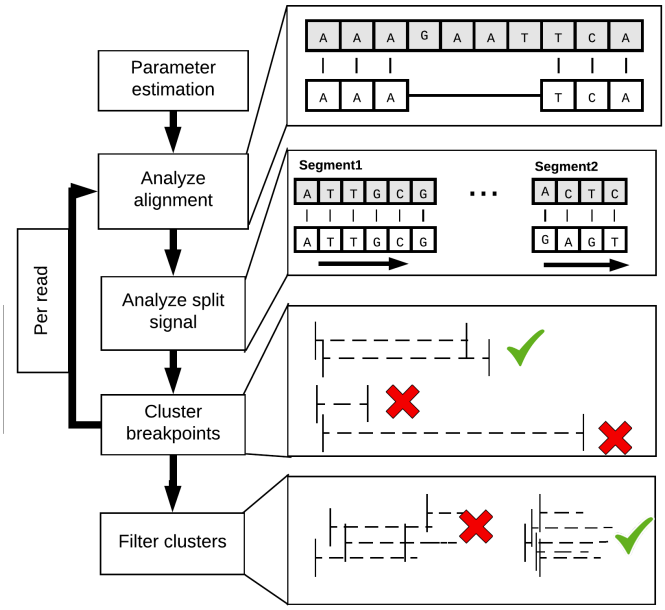


Figure 4.6: Overview of the SV calling algorithm implemented by Sniffles.

as small indels can be spanned within a single alignment, but large or complex events lead to split-read alignments. The major advance of Sniffles is its ability to filter false SV signals from the noisy reads. As with other variant detectors, minimum read support (default: 10 reads) is a critical feature, but Sniffles also considers the consistency of the breakpoint position and size. In addition, it can perform read-based phasing of SVs and report adjacent or nested events in the output VCF (Variant Call Format) file.

4.2.3 Benchmarking NGMLR and Sniffles

To validate NGMLR and Sniffles we performed a series of evaluations using simulated and genuine read data. First, we modified existing reference genomes by adding simulated SVs and created simulated long reads based on these modified genomes. Next, we aligned the simulated reads to the initial reference genome, called SVs and compared these calls to the set of simulated SVs (truth set). This approach allowed us to evaluate how accurate the read alignments are, how many

of the simulated SVs we are able to call correctly, and how many of our calls are false positives.

Benchmarking using simulated SVs and simulated reads

We used the SURVIVOR toolkit (Jeffares et al., 2017) to simulate insertions, inversions, deletions, duplications, and translocations. Furthermore, we extended SURVIVOR to introduced nested SVs such as an inversion flanked by two deletions or a tandem duplication which has one copy inverted. For the evaluation, each breakpoint is treated as a separate event, e.g., a caller must call the inversion and the two deletions separately to be correct. To simplify analysis, we simulated datasets that hold only one specific type of SVs and a specific average length of the SV. Specifically, we simulated a size range of 100bp, 250bp, 500bp, 1kbp, 2kbp, 5kbp and 10kbp, and for each data set we simulated a total of 20 SVs of the same type. For translocations we randomly chose a region of a defined size (e.g., 250bp) to be swapped with a region of the same length from a different chromosome. To simulate INVDEL variants we simulated an inversion flanked with two deletions that are 10% of the total size. Thus, INVDEL data set consists of 20 inversions and 40 deletions.

Read simulation: To simulated PacBio and Oxford Nanopore reads from our modified genomes we extended SURVIVOR with an error profile generator and read simulator that can generate long reads based on an error profile and a specified coverage. Given alignments from a genuine long read dataset the error profile generator module scans all aligned reads and records the ratio of reads showing a deletion, insertion, match or mismatch at each read position. Furthermore, it records the number of reads that overlapped with each position. Figure 4.7a and b show the error profiles for Oxford Nanopore and PacBio data, respectively. We note that the error profiles become highly variable towards the end because of the limited numbers of reads available for sampling. For this analysis we used the BWA-MEM alignments from a PacBio HG002 dataset provided by the Genome in a Bottle Project (GiaB) (Zook et al., 2014) and an Oxford Nanopore NA12878

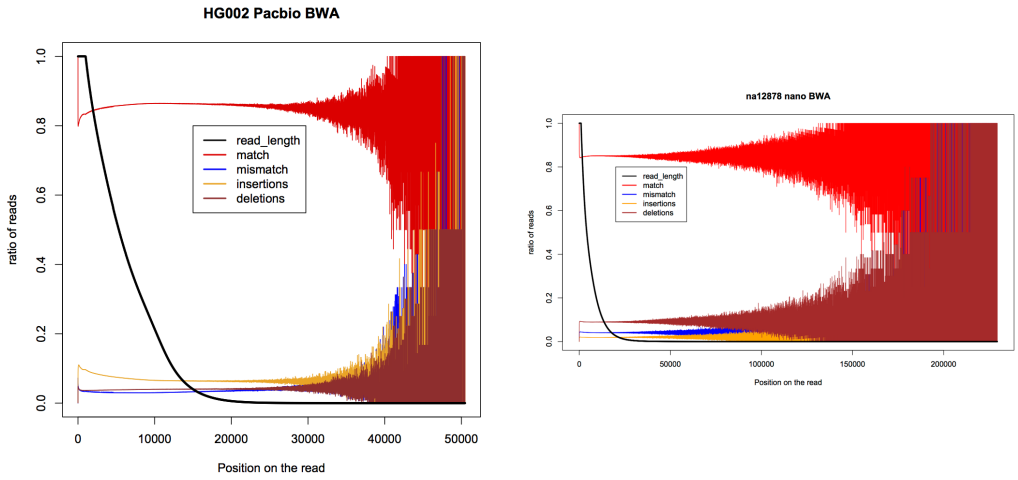
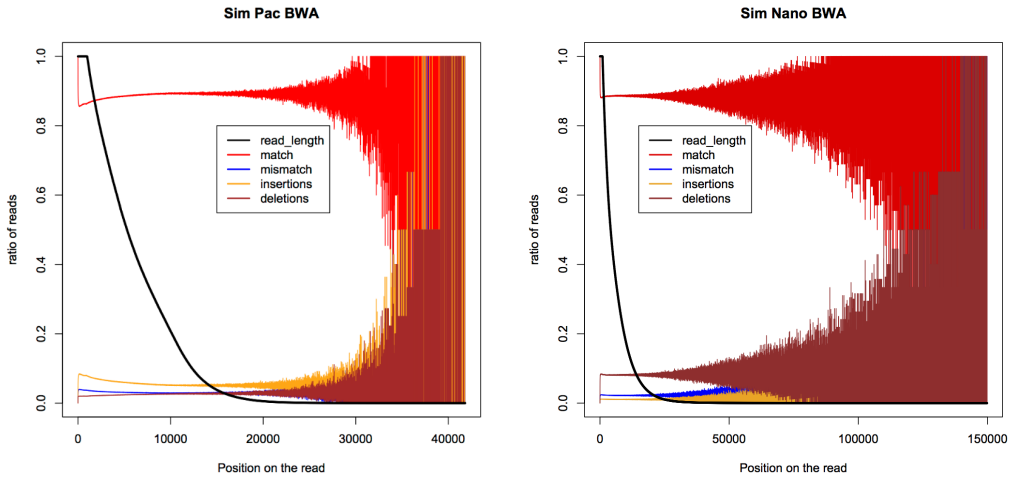


Figure 4.7: Visualisation of an empirically determined per read position error profile for (a) PacBio and (b) Oxford Nanopore reads. For read positions $>$ 20kb in PacBio reads and $>$30kb in Nanopore reads the error profiles are unreliable as there are not enough reads of these lengths in the dataset.

dataset (Jain et al., 2018) requiring a minimum alignment length of 1kb. Using error profiles derived from those alignments we simulated reads from the modified reference genome. Briefly, we first compute the required number of reads for a given level of read coverage given the median read length. For each read, chromosome and starting position are chosen randomly. The position on the chromosome and the length of the read defines the sequence of the read. If more then 10% of the read sequence are N's we discard the read and chose a new random starting location. The read subsequence is then altered according to the error profile shown in Figure 4.7. A limitation of this approach is that it only allows us to introduce insertion and deletion of length 1 for simulating sequencing errors. However, we found this to sufficiently capture the main characteristics of sequencing error for all practical purposes.

For each data set we simulated 347,538 PacBio-like reads of length 21kbp and 662,943 Nanopore reads. Furthermore, we simulated 39.8 million Illumina like 100bp paired-end reads with an insert size of 500bp using Mason (Holtgrewe, 2010). Figure 4.8 a and b show the profiles computed from our simulated data PacBio and Oxford Nanopore like reads.



(a) **(b)**

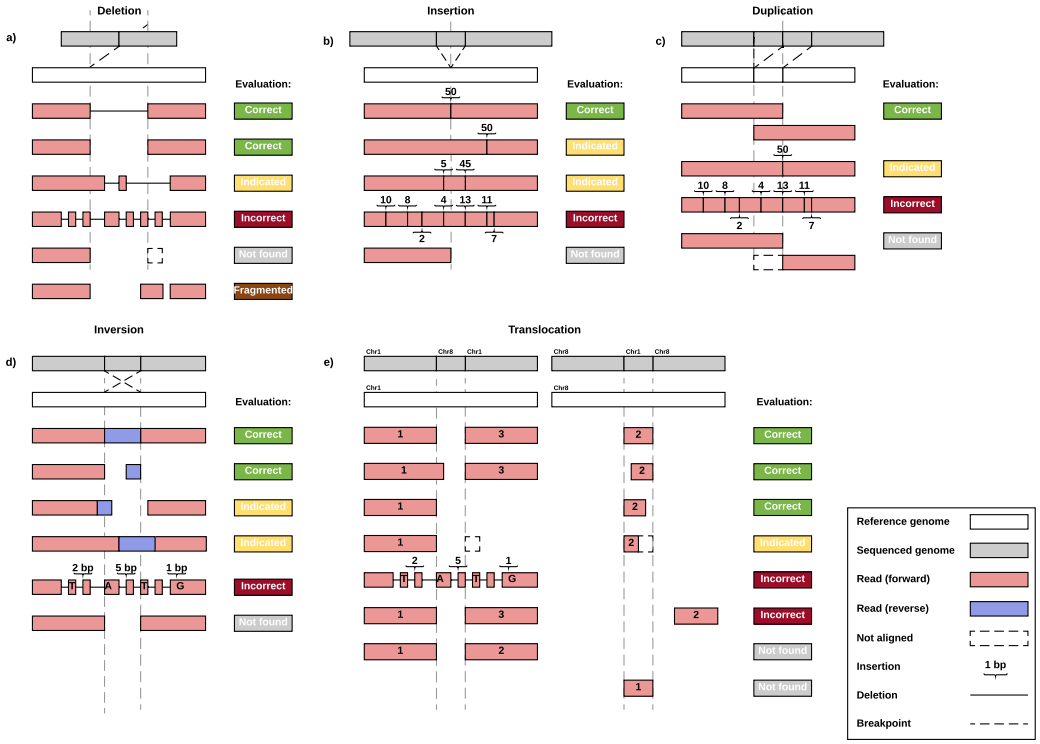
Figure 4.8: Per read position error profile for our simulated (a) PacBio and (b) Oxford Nanopore datasets. For read positions $>$ 20kb in PacBio reads and $>$30kb in Nanopore reads the error profiles are unreliable.

Evaluation of read mappings and SV calls: All simulated reads were mapped to the human reference genome using BWA-MEM, BLASR, GraphMap, MECAT and NGMLR. Reads that overlap or map in close proximity to a simulated SV were extracted from the BAM files. The extracted reads were divided into six categories (see Figure 4.9):

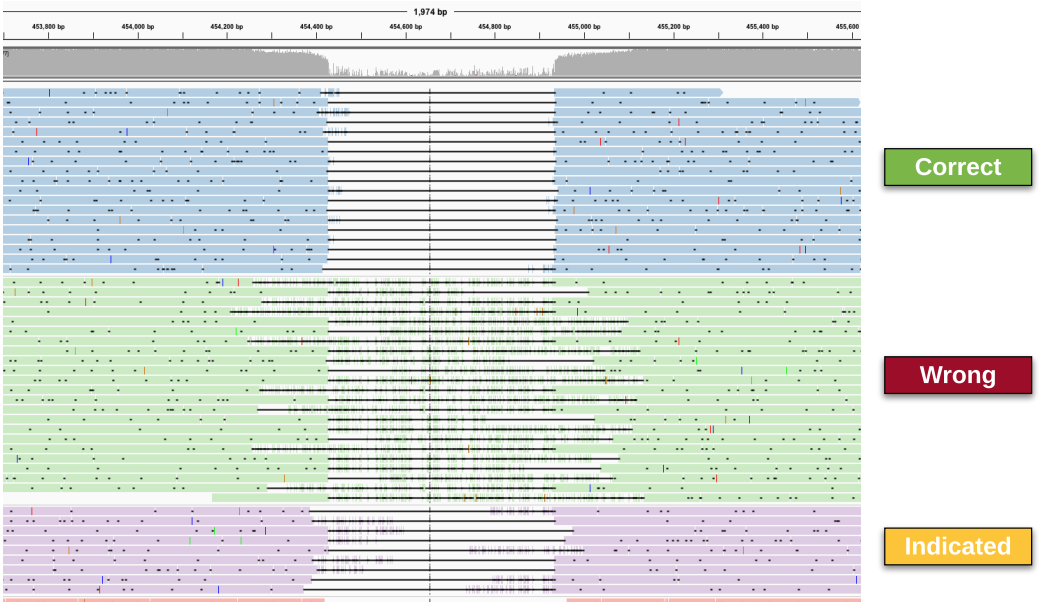
Correct Read mappings are considered correct (illustrated in green) if they allow to fully identify the SV they cover. To fall into this category, read mappings have to cover all parts of the SV that are required for identification, e.g., a read mapping to an inversion has to cover the inverted part of the genome as well as the non-inverted part(s) flanking the inversion. Furthermore, correct mappings have to be split at the simulated breakpoints (+/- 10bp) of the SV.

Indicated Read mappings that indicate the correct SV (e.g., a duplication that is represented as an insertion) or mappings that show the correct SV but do not show the exact borders are considered partially correct (yellow).

Incorrect Mappings that indicated the wrong SV or contain a significant portion of mapping artefacts (e.g., not simulated mismatches) ($>$ 10% of the SV length)



(a)



(b)

Figure 4.9: Evaluation of simulated reads. All reads are divided into six categories depending on how well they capture the SV they overlap. (a) Schematic visualisation of example reads for all evaluation categories and all evaluated SV types. (b) IGV screenshot of read alignments coloured and grouped by evaluation category.

are considered incorrect (red). E.g., a read that is forced to align through an inversion.

Not found Reads that do not indicate any SV and do not contain randomly aligned base pairs (i.e. noisy regions) fall in the category not found (grey).

Fragmented Reads that are split into more parts than required to cover the underlying SV are classified as fragmented (brown).

No aligned Reads that are supposed to map across the SV but are not mapped (white)

For all simulated SV types, sizes and mappers, we counted what percentage of reads fell into the six categories and visualised the result as barplots.
Furthermore, SVs were called from the read alignments using Sniffles, PBHoney, Lumpy, Manta and Delly and evaluated in a similar manner:

Precise SVs are considered "precisely detected" if their breaks points are reported within 10bp up or downstream of the simulated (true) position and the type of SV (e.g., insertion, deletion) is correctly reported (green).

Indicated A caller "indicates" the right call if it reports an SV within 1kbp up or downstream of the true location. Furthermore, the reported type of SV does not have to be correct (yellow).

Not detected An SV is considered not detected if there is no call found within 1kbp up or downstream of the true location (red).

False positive Furthermore we consider calls that were found but not simulated as false positives (brown).

Similarly, to the read mapping evaluation, we summarized all results and visualised them as a colored barplot.

Benchmarking using genuine read data and simulated SVs

After establishing Sniffle's and NGMLR's performance on simulated that we next used read data from genuine NA12878 PacBio (Zook et al., 2014) and Oxford

Nanopore (Jain et al., 2018) datasets to verify our results. Since the true structural variants are unknown in this genome, we had to alter the reference sequence to introduced artificial SVs. We simulated:

1. 140 insertions by deleting regions from the reference
2. 140 deletions by adding new sequence to the reference
3. 140 inversions by inverting parts of the reference
4. 140 balanced translocations by exchanging regions of the same length from two different chromosomes creating 280 translocation events

We did not attempt to simulate tandem duplications, as this would require detecting and modifying existing tandem duplications in the reference. The mean indel and inversion size was 1.6kb. After aligning the genuine PacBio and Oxford Nanopore reads to these simulated references and calling SVs we used the same approach to evaluate read alignments and SV calls as for simulated reads (see previous paragraph).

Benchmarking using published SV calls from genuine reads datasets

Evaluation using simulated SVs can give an upper bound for SV calling performance. However, it cannot capture the full complexity of naturally occurring SVs. To date no full SV truth-sets or gold standard call-sets exists for NA12878 (or any other human sample). Thus, to investigate NGMLR and Sniffle's ability to call genuine SV we compared to existing (incomplete) call-sets. To this end, we downloaded the following published SV callsets for NA12878:

1. One calls set[1] from the Platinum genomes project and the 1000 genomes project for NA12878 currently hosted on dbGaP: phs001224.v1.p1[2]. The data set consisted of 1802 deletions.
2. callsets provided by GiaB from 7 different SV callers[3].

[1] NA12878.wgs.illumina_platinum.20140404.svs_v2.vcf

[2] `https://www.illumina.com/platinumgenomes.html`

[3] `ftp://ftp-trace.ncbi.nih.gov/giab/ftp/data/NA12878/NA12878_PacBio_MtSinai/`

Next, we called variants for a PacBio and an Oxford Nanopore NA12878 datasets using NGMLR and Sniffles using an unaltered version of the human reference genome. The resulting calls represent the true genetic variation between NA12878 and the human reference. Using the SURVIVOR toolkit and allowing for a 1kbp distance between two calls we computed how many SV the callsets had in common and how many where unique to different sequencing technologies and analysis pipelines. Based on the results we then investigated differences between sequencing technologies specifically for indels and translocations calls.

Assessing discrepancies between indel calls from different platforms: First we checked for short read support for deletion calls from our PacBio and Oxford Nanopore datasets. To this end we looked for changes in short read insert sizes around indel calls from Sniffles. We customized the script [4] provided in the Lumpy (Layer et al., 2014) package to obtain the mean and standard deviation of insert sizes across the entire Illumina data set. The mean insert size was 311.70. Using the SURIVOVR toolkit we converted the insertion and deletions with a length of 50bp to 3kbp to a BED file containing the chromosome, start and stop coordinates of the PacBio-based or Oxford Nanopore-based SVs. Note that the stop coordinate is with respect to the reference genome, so does not contain the length of the insertion. We then identified Illumina read pairs mapping within 300bp up and downstream of the start and end breakpoint of an insertion or deletion. For each of the insertions and deletions we computed the mean and standard deviation of the insert size of all the spanning short read pairs. These were then tested for a significant deviation compared to the global average insert size using a two-sided, one sample t-test. We considered a p-value < 0.01 as significant and interpreted it as indication for the short read pairs supporting the SV call from the PacBio or Oxford Nanopore datasets.

Assessing discrepancies between translocation calls from different platforms: To investigated differences between short and long read translocation calls, using the SURVIVOR toolkit, we first identified translocation calls called in at least

[4] pairend_distro.py

2 of our Illumina callsets. Through manual inspection of the short read alignments, we found many translocation calls to overlapped with short insertions in low-complexity regions (e.g., di-nucleotide repeats). To quantify how many of the short-read translocation calls overlapped with insertions and other SV called by Sniffles, we reran Sniffles with less stringent settings using `-l 10` (to obtain all SVs of length 10bp or larger) as well as `-s 5` (to require only 5 supporting reads for an SV to be called) and converted these calls to the BED format. In the BED file, each short-read translocation call was represented by two 400bp intervals centered around its break points. Next, we split the Sniffles calls into 5 separate VCF files, where each file contained only one of the following SV types: translocation, insertion, duplication, deletion, inversions. We used bedtools (Quinlan and Hall, 2010) to identify if at least one of the break points of a short-read translocation call, overlapped with SVs in one of the five Sniffles VCF files. If a short-read translocation call overlapped with more than one Sniffles call, we counted only the first overlap we found.

For the remaining short-read translocation calls that did not overlap with any Sniffles call, we checked whether they overlapped with a repeat annotation or a region with substantially increased read coverage as this could indicated missmapping of the short reads. To this end, we computed the coverage for the 400 bp regions centered around the translocation break points using bedtools multicov. As a baseline we used the coverage of randomly shuffled 400bp intervals (bedtools shuffle). We considered a translocation as falling in a high coverage region, if at least one of its breakpoints shows a higher coverage than all the random intervals tested.

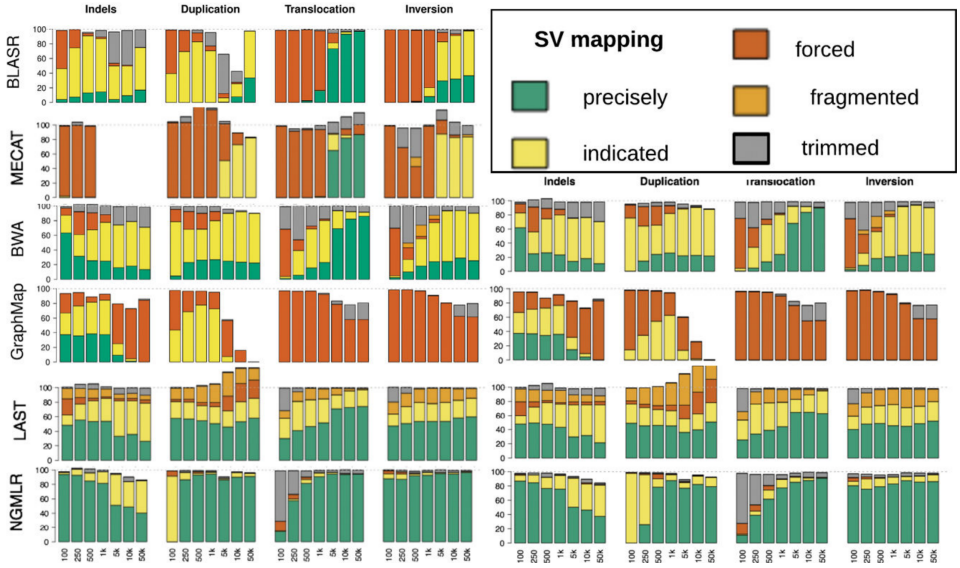


Figure 4.10: Read mapper evaluation with simulated data. In each plot, the x-axis indicates the size of the 840 simulated SVs. We simulated PacBio-like (left) and Oxford Nanopore-like reads (right) and determined whether alignments were precise, indicated, forced, unaligned, or trimmed but not forced to wrongly aligned across the SV. For the SV analysis (bottom), we used the same alignments as before and distinguished among precise, indicated, not indicated, and false positive calls.

4.3 Results & Discussion

4.3.1 Evaluation of NGMLR and Sniffles using simulated SVs

Simulated reads

We simulated 50X PacBio-like and 50X Oxford Nanopore-like read datasets from two human chromosomes (chr21 and chr22). In the simulation, we included a total of 840 homozygous SVs consisting of equal numbers of indels, duplications, balanced translocations, and inversions ranging from 100bp to 50kbp in size (see Method section for more details). Figure 4.10 (left) summarizes the results when evaluating read alignments from NGMLR, BWA-MEM, BLASR, GraphMap, LAST and MECAT. Each stacked bar represents one data set consisting of 20 SVs of a certain type and length. Across all SV types, NGMLR outperformed the other mappers with an average of 80.32% precisely aligned reads versus 26.03% for the second-

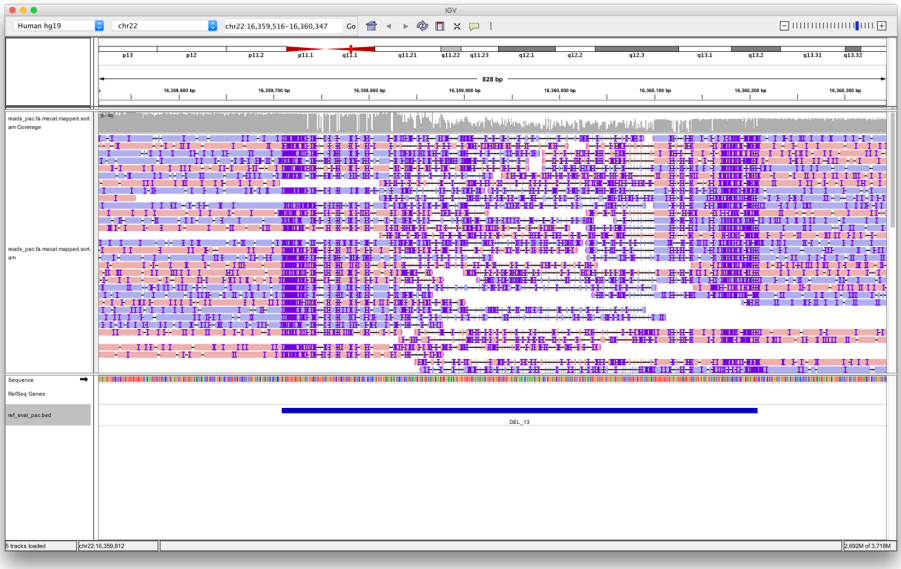


Figure 4.11: IGV screenshot of incorrect MECAT alignments spanning a deletion.

best tool. When counting all alignments that could aid SV detection (precise and indicated), NGMLR showed the highest performance with an average of 91.83% versus 69.17% for the next closest competitor. Except for NGMLR, essentially all the other aligners showed a high number of alignment artifacts and overall poor performance when aligning reads spanning SVs (see Figure 4.11 for an example from MECAT).

Next, we compared the performance of NGMLR, BWA-MEM, GraphMap and LAST when mapping simulated Oxford Nanopore-like reads. BlasR and MECAT were excluded, as they are only applicable to PacBio reads. Again, NGMLR outperformed the other mappers in terms of precisely aligned reads (72.42% vs 24.90% for the second-best), or when considering both precise and indicating alignments (88.57% versus 67.95%) (Figure 4.10 right). GraphMap performed poorly on these data, with on average only 18.19% of reads aligned precisely or indicating the SV. 61.13% of GraphMap's alignments showed aliment artifacts which obfuscated the SV spanned by the read. Please see Supplementary Table B.1 a full list of all evaluation results.

Next, we evaluate the performance of Sniffles compared to other short and long

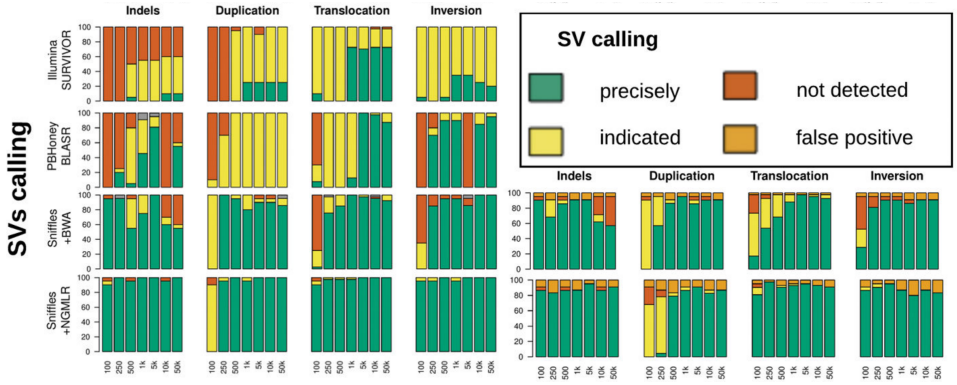


Figure 4.12: SV caller evaluation with simulated data. In each plot, the x-axis shows the size of the simulated SVs. We distinguished between precise, indicated, not detected, and false positive calls.

read SV detection tools using the PacBio-like and Nanopore-like alignments reported above (see Methods). We were able to use Sniffles with either NGMLR or BWA-MEM. The output formats of the other aligners are not compatible with Sniffles. We also extended the analysis to include simulated short reads. We called SVs for the short reads using an ensemble calling method implemented in the SURVIVOR toolkit. SURVIVOR aggregates the outputs from several short read SV caller (Lumpy, Manta and Delly) and excludes variants reported by only a single caller. This approach was shown to increases specificity without sacrificing sensitivity (Jeffares et al., 2017).

Using PacBio data the combination of Sniffles and NGMLR performs best with an average of 94.20% precisely detected SVs and a false discovery rate (FDR) of 0.00% (Figure 4.12 left). We found that the most problematic class of SVs was short (100bp) tandem duplications, as they are identified as insertions rather than tandem duplications. Hence, we classified them as indicated and not as precisely called. The second-best result was achieved using Sniffles with BWA-MEM alignments, with on average 79.11% precisely detected SVs and a 0.68% FDR. PBHoney, which relies on BlasR alignments, precisely detected only 32.58% of simulated SVs and missed 25.18%. Most of the 40.73% indicated SVs from PBHoney came from misinterpreting tandem duplications as insertions. For the short-read analysis, SURVIOR detected 18.81% of the simulated SVs precisely and indicated

another 57.89%. This matches previous findings for short-read based SV analysis (Hartigan and Hartigan, 1985, Jeffares et al., 2017). Only the the FDR was lower (0.17%) then usually observed with short reads due to the consensus-based SURVIVOR analysis we used. See Supplementary Table B.2 for more details.
Finally, we benchmarked the performance of Sniffles using BWA-MEM and NGMLR on the Oxford Nanopore-like reads (Figure 4.12 right). Using Sniffles with NGMLR, 82.25% of SVs were precisely identified, compared to 76.35% for BWA-MEM and Sniffles. Due to the higher rate of sequencing errors in the Oxford Nanopore-like data, Sniffles using either aligner showed a higher FDR calling 1-4 false positive events per data set.

Genuine long human reads

Evaluations using simulated reads establish a baseline of performance but may not capture the full complexity of real sequencing data. To benchmark more realistic datasets, we next analyzed genuine PacBio and Oxford Nanopore reads from the well-studied NA12878 human cell line. Since a complete truth set of SVs is not available for this genome, we modified the human reference genome to introduce 700 simulated homozygous SVs (see Methods).
In this analysis, we can only evaluate how well the alignments capture our simulated SVs. We cannot quantify the number of false positive calls introduced by alignment artifacts as they would be impossible to distinguish from true SVs present in NA12878. NGMLR showed a clear improvement over BWA-MEM (58.65% vs 32.35%) for precisely aligned reads across the SVs (Supplementary Table B.3). The shorter average length of the genuine reads limited the number of reads that could be aligned precisely. For example, if an insertion is longer than the read length, the read can only indicate the SV. When summing up precise and indicated calls, NGMLR achieved a substantially better result than BWA-MEM (76.96% vs 49.21%). Furthermore, NGMLR considerably reduced the number of reads showing alignment artifacts compared to BWA-MEM (3.01% vs 24.21%). Using the Oxford Nanopore reads we observed a similar trend with NGMLR giving the most

precise alignments (51.56% vs. 27.35%) with the lowest percent of alignment artifacts (5.94% vs. 29.15%).
Using the long reads alignment discussed in the last paragraph and alignments of 50X genuine Illumina sequencing data (Eberle et al., 2017), we next benchmarked the available SV callers (Supplementary Table B.4). Sniffles and NGMLR achieved the highest rate of precisely called SVs with 95.14% using PacBio and 88.29% using Oxford Nanopore reads, respectively. In contrast, the short-read based SURVIVOR analysis had a much lower percentage (76.57%) of precise and indicated SV calls.

4.3.2 Trio-based analysis of genuine SVs

So far, using simulated SVs we have shown NGMLR and Sniffles to be the most accurate of all tested long-read SV calling tools. Next, we investigated the performance of long-read based SV calling using NGMLR and Sniffles and how it compares to conventional short-read based calls using real SVs. Since no SV truth sets exist to evaluate precision and recall directly, we used sequencing data from trios. When sequencing parents and offspring we can use assumptions based on the expected Mendelian inheritance pattern to evaluate SV calling. We will use two assumptions. First, all homozygous SVs identified in the parents should be present in the offspring as well (as a heterozygous or homozygous SV). This allows us to estimated recall (we will call it parental recall here). Second, since the rate of spontaneously occurring SVs is expected to be low, the vast majority of SVs detected in the offspring (both heterozygous and homozygous) should be present in at least one of the parents. This allows us to identify missing calls (false negatives) in the parents or additional calls (false positives) in the offspring. We will call these calls discordant calls.

Arabidopsis thaliana Trio analysis

First, we looked at a trio of *Arabidopsis thaliana* genomes previously sequenced with PacBio and Illumina by Chin et al. (2016). The trio consisted of two inbred

Dataset	Technology	Coverage	Read length	Total SVs
Col-0	PacBio	127x	6,482	355
CVI	PacBio	123x	6,073	9,652
Col-0xCVI	PacBio	155x	11,206	11,935

Table 4.1: Number of SVs detected in *Arabidopsis thaliana* PacBio data.

cultivars called Col-0 and CVI and a F1 hybrid of the two (Col-0xCVI). Using NGMLR and Sniffles, we identified 355 SVs in Col-0 when comparing it to the standard *Arabidopsis thaliana* reference genome version 10 (TAIR10). For CVI, NGMLR and Sniffles detected 9,652 SVs (Table 4.1). 42 and 6,679 of these calls were homozygous in Col-0 and CVI respectively and thus expected to be present in Col-0xCVI. When comparing our Col-0xCVI calls to the homozygous calls from Col-0 we found that 4 of the homozygous SVs were not identified in Col-0xCVI which equals a parental recall of 90.4%. On closer inspection we found that one missed insertion was reported as 47bp in Col-0xCVI vs. 53bp in Col-0, and similarly a deletion was reported as 48bp in Col-0xCVI vs. 53bp in Col-0. Thus, these SVs were initially not found due to our minimum length cut-off of 50bp. Similarly, NGMLR and Sniffles were able to detect the remaining two SVs - another deletion and a duplication - in the Col-0xCVI when we reduced the read support required to call a SV. The deletion was supported by only 4 reads and the duplication by only 3 reads. When looking at the CVI calls, NGMLR and Sniffles initially missed 370 SVs (94.5% parental recall). 159 of these were present but lacked a sufficient number of supporting reads in Col-0xCVI; 101 had slightly different SV sizes reported which were below the minimum size; 43 were interpreted as different SV types; and 50 were caused by spurious read alignments that could easily be filtered out (see Appendix B.2). After considering these factors, only 17 SVs present in the CVI data set were missed by NGMLR and Sniffles (parental recall of 99.9%). Next, we looked at the discordance rate. NGMLR and Sniffles detected 767 SVs that were unique to Col-0xCVI, resulting in a discordance rate of 7.22%. Upon closer investigation, we found that most of these calls were caused by a shorter average read length in the Col-0 and CVI datasets. The shorter reads caused certain repetitive

Dataset	Technology	Coverage	Read length	Total SVs
HG002 (son)	PacBio	69x	8,540	19,131
HG002 (son)	Illumina	80x	148	10,822
HG003 (father)	PacBio	32x	6,284	11,964
HG003 (father)	Illumina	80x	148	11,395
HG004 (mother)	PacBio	30x	7,285	10,463
HG004 (mother)	Illumina	80x	148	8,901

Table 4.2: Number of SVs detected in PacBio and Illumina data from a human trio.

regions to have lower coverage than expected after mapping quality filtering. When adjusting our read support parameter to a minimum of 5 reads, the inconsistency rate dropped to 3.4%. When allowing for larger distances SV calls in the parents and the offspring (10kbp) the discordance rate dropped further to 1.2%.

Genome-in-a-Bottle Human Trio Analysis

To evaluate the performance NGMLR and Sniffles for human datasets we used PacBio and Illumina data of the Ashkenazim trio provided by the GiaB consortium. NGMLR and Sniffles reported 5,244 and 5,964 SVs as homozygous in the father and mother, respectively. 93.84% and 94.01% of these SVs were found in the son as well. In contrast, when using the Illumina-based SURVIVOR calls, we identified only 1,586 and 1,668 homozygous SVs for father and mother, respectively. Note, the coverage in the Illumina data was $>$2.5 times higher than for the long-read datasets (Figure 4.2). Of the homozygous calls, 89.66% and 87.82%, respectively, were found in the son as well. The discordance rate was 5.8% for NGMRL and Sniffles and 21.9% for the short-read callset (Supplementary Table B.5e). Interestingly, when splitting the discordant calls by SV type we found that most of the discordant calls from short reads were translocation calls (1,550 vs. 90 for NGMLR and Sniffles).

Source	Technology	Deletions	Duplications	Insertion	Inversions	Translocations
NGMLR+Sniffles	PacBio	6,734	606	7,880	160	119
NGMLR+Sniffles	Oxford Nanopore	19,074	761	6,376	334	112
SURVIVOR	Illumina	3,744	553	0	731	2,247
1k genomes project	Illumina	1802	0	0	0	0

Table 4.3: Number and types of SVs detected in human NA12878 datasets for different sequencing technologies.

4.3.3 Comparison of Illumina, PacBio and Oxford Nanopore SV calling results

All sequencing platforms come with different limitations for SV calling. To quantify differences and biases in SV calling for the different sequencing platforms we compared Sniffles (using NGMLR alignments) and SURVIOR calls for three NA12878 datasets (PacBio (Zook et al., 2014), Oxford Nanopore (Jain et al., 2018) and Illumina (Zook et al., 2014)) with a short-read deletion-only call-set from the 1000 genomes project and a long-read call-set provided by the GiaB consortium. NGMLR and Sniffles identified a total of 15,499 SVs using PacBio reads and 26,657 SVs using the Oxford Nanopore reads. SURVIVOR reported 7,275 SVs (Table 4.3) for the short read dataset. Together, the five callsets yielded a total of 40,601 SVs. The majority (24,392) of the identified SVs were present in only one callset (see Figure 4.13), whereas 16,209 SVs were identified in two or more callsets. Of the 15,499 PacBio calls, most (94.8%) were confirmed by Oxford Nanopore, Illumina, or the other existing call-sets. Oxford Nanopore had a substantially lower concordance, as NGMLR and Sniffles reported 11,433 calls unique to Oxford Nanopore. 10,977 (96.0%) of these calls were deletions and the majority (92.9%) overlapped homopolymers or other simple repeats. In contrast, the 773 calls found only by PacBio were mainly insertions (66.5%), and only 323 (41.8%) overlapped homopolymers or repeats. This systematic bias for deletions in the Oxford Nanopore data is most likely due to base-calling errors, also reported by Jain et al. (2018). The majority of these artifacts are small deletions. When we increased the minimum SV size to 200bp, NGMLR and Sniffles reported only 38.6% of the SV calls within homopolymers and low-complexity regions (Supplementary Table

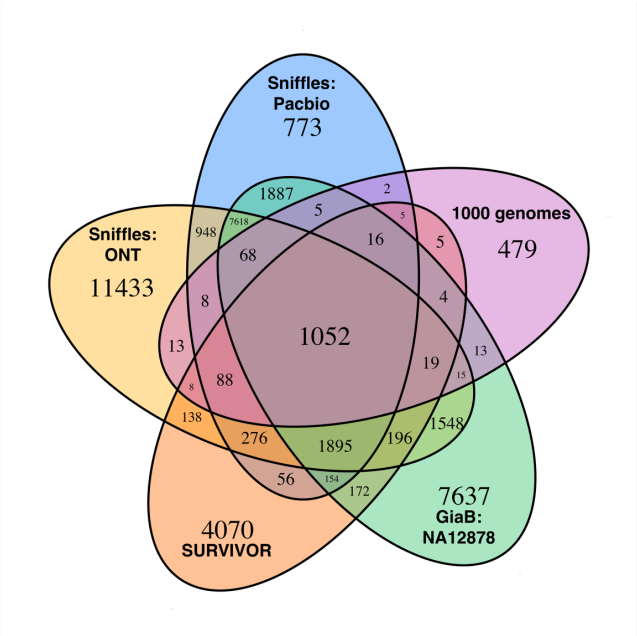


Figure 4.13: Overlap between SV calls from different call-sets for NA12878.

B.6).

Illumina-based SV calling had low concordance with alternative approaches. 49.7% of Illumina calls were unique to the technology. Matching the results from our trio analysis, the majority (54.1%) of unique calls were translocations. Deletions on the other hand were underrepresented in Illumina call-sets. We observed 40-50 % fewer deletions calls compared with long reads. The biggest limitation however is precise calling of insertions. Only one of the SV callers (Manta) used by SURVIVOR supports insertion calling which relies on read-pair distance changes and thus can neither report exact break points nor the inserted sequence. Since SURVIVOR requires two short read callers to support a call, insertions are not reported.

Investigation of unique short-read versus long-read SV calls

The two most striking difference between long and short read SV call-sets were the high number of insertions and deletions called with long reads and the high number of translocations called in short reads. We first looked at small insertion (50 - 300bp) and deletion (50 bp - 3 kbp) calls from Sniffles to see if we can find

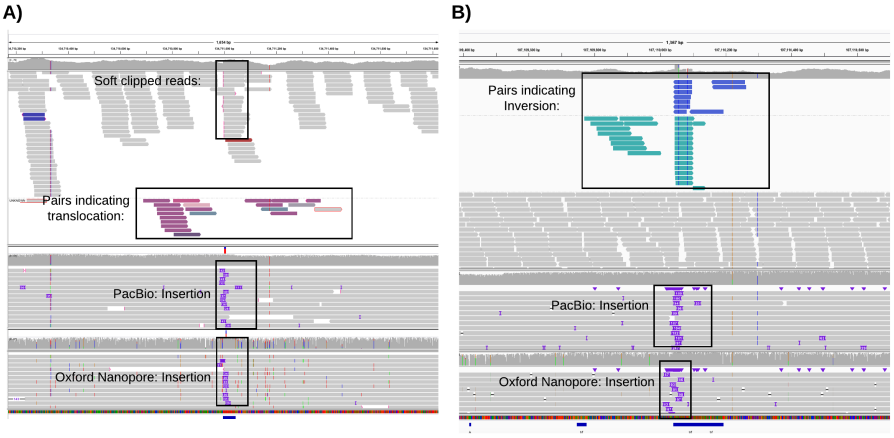


Figure 4.14: Systematic error in short-read based SV calling. (a) Example of a putative translocation identified in short-read data (top alignments) that overlaps an insertion detected using PacBio (middle) or Oxford Nanopore sequencing (bottom). (b) Example of a putative inversion identified in the short-read data (top) that overlaps an insertion detected in both PacBio (middle) and Oxford Nanopore reads (bottom).

support from short reads. We used the distance between Illumina read pairs that span Sniffles call as orthogonal evidence. To this end, we used the compression-expansion statistic (Zimin et al., 2008) as an unbiased measure of the Illumina read-pair placements near predicted indels (see Methods). True insertions should cause BWA-MEM to map read pairs closer than expected, and deletions farther away, with respect to the average Illumina insert size of 311 bp observed genome-wide. Using the Illumina data and a P value threshold of 0.01 (two- sided, one-sample t-test), we confirmed 3,415 (PacBio) and 3,879 (Oxford Nanopore) deletions and 2,685 (PacBio) and 1,703 (Oxford Nanopore) insertions reported by NGMLR and Sniffles (Supplementary Table B.7).

Next, we investigated the large number of translocations reported in the Illumina-based consensus calls. We found a large overlap (48.9%) between Illumina-based translocation calls and insertion calls from NGMLR and Sniffles supported by both long-read technologies. Figure 4.14a shows a representative example of a long-read insertion call overlapping with a low-complexity region which causes short reads to be mismapped. These mismapped reads were then misinterpreted as reads that span a translocation by the SV caller even when excluding low mapping quality

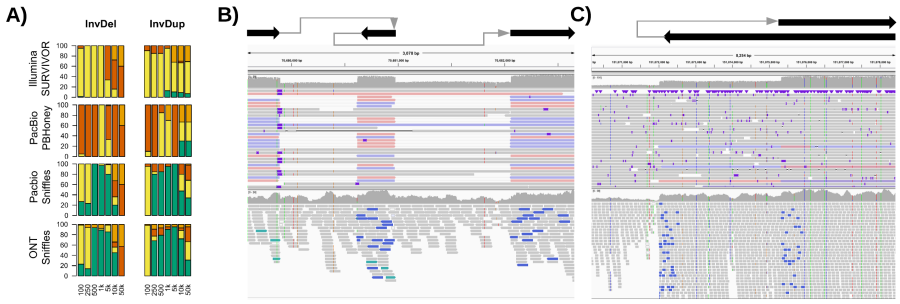


Figure 4.15: Nested SV calling in simulated data and the SKBR3 cancer cell line. (a) Evaluation of NGMLR and Sniffles with simulated data to identify nested SVs. (b) A 3-kb region including two deletions flanking an inverted sequence is clearly visible and was detected by NGMLR and Sniffles (top) but was not detected using short-reads (bottom). (c) The start of an inverted duplication. Breakpoints were reported by Sniffles as the start of an inverted duplication (top) but were not correctly detected using short-reads (bottom).

($<$ 20) reads. In total, 1,869 (83.2%) of the Illumina-based translocation calls, overlapped with long-read insertion calls (48.9%). An additional 8.9% overlapped with short deletions calls that led to similar mapping artifacts (Supplementary Table B.7). 404 (18.0%) of the remaining Illumina-based translocation calls were located in low-complexity regions where unique mapping of short reads is difficult. 141 (6.3%) overlapped with regions that showed abnormally high coverage in the short read data but no increase in coverage in the long-read data. Inversions showed a similar pattern: 60% of calls overlapped with a different SV type identified using long reads (Fig. 4.14b) or aligned to low-complexity regions of the genome. In summary, the majority of PacBio-based indel calls from NGMLR and Sniffles were validated by either Oxford Nanopore or Illumina paired-end reads. In contrast, the majority of calls unique to the Illumina-based methods were false positive calls, especially translocations caused by mismapped reads in low complexity regions.

4.3.4 Detection of complex SVs

In addition to single SVs that occur in isolation, multiple SVs can occur at the same time (nested SVs) or in close proximity and form more complex rearrangements.

Known examples of nested SVs are inverted duplications (INVDUPs) and inversions where parts of the inverted sequenced are deleted (INVDELs). INVDUPs have been associated with Pelizaeus-Merzbacher disease (Shimojima et al., 2012) and a number of other conditions (Carvalho et al., 2011, Beri et al., 2013). INVDELs haven been observed in Hemophilia A patients (Mühle et al., 2007). To investigate the performance of NGMLR and Sniffles for calling nested SVs, we simulated 280 nested SVs of different sizes and types in the human genome, along with simulated PacBio-like, Oxford Nanoporelike, and Illumina-like reads (Fig. 4.15A). Using PacBio data Sniffles called 67.9% of the nested SVs precisely and partially detected another 22.9%. With Oxford Nanopore-like reads, Sniffles detected 67.3% of simulated SVs precisely and 21.9% partially. Neither PBHoney using PacBio reads nor SURVIVOR calls from short reads were able to accurately identify these nested SVs (2.6% and 2.7% precisely called). PBHoney missed most of the SVs completely (62.7%). Using short reads, most of the SVs were only partially detected (71.2%). See Supplementary Table B.2 for a full list of all evaluation results.

To further investigate calling of nested and complex SVs using real data, we examined a PacBio dataset for the SKBR3 breast cancer cell line (Nattestad, 2017). NGMLR and Sniffles revealed 15 gene fusions caused by complex SVs with as many as three rearrangements happening in close proximity. Figure 4.15B shows an example where short read alignments indicated an inversion, but the poor break point resolution made it impossible to detect and connect all breakpoints. In contrast, NGMLR and Sniffles detected an INVDEL and an INVDUP, as well as reads that span both calls and thus fully resolved this complex rearrangement.

4.3.5 How much coverage is required?

Finally, we assessed how much coverage is required to detect SVs using long read sequencing. This is an important consideration as long-read sequencing is still more expensive than short read sequencing (Goodwin et al., 2016). Based on a naive simulation (see Methods), about 10X coverage should be sufficient to infer all SV breakpoints (100% recall) using 10-kbp reads in a human genome, whereas about

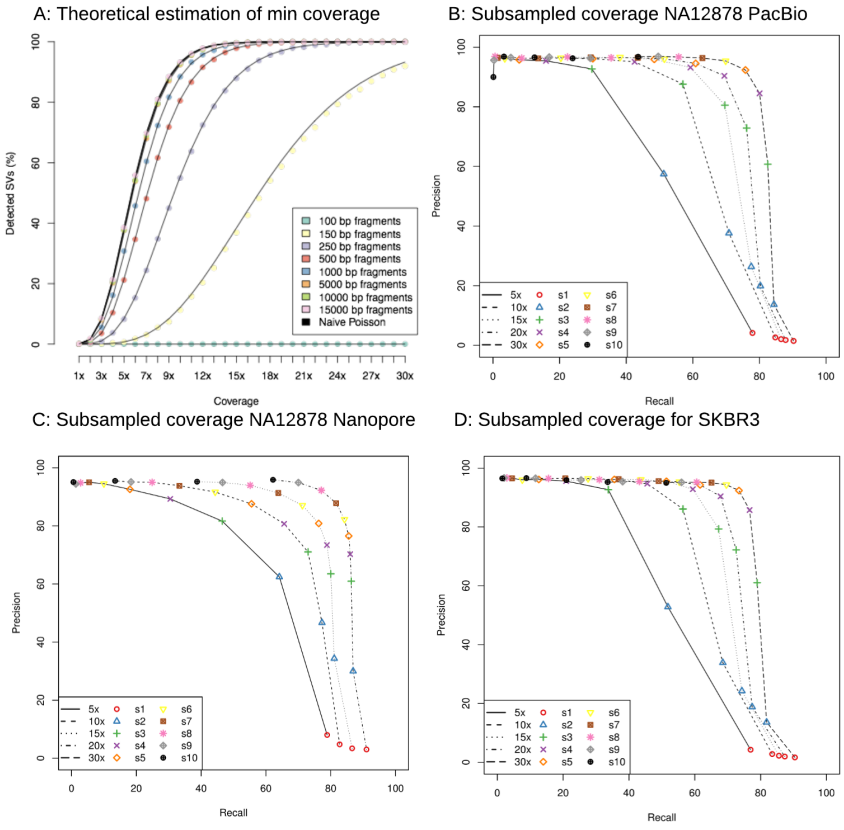


Figure 4.16: (a) Theoretical assessment of recall versus coverage for different read lengths requiring 50-bp overlap of each breakpoint for SV calls. (b) Subsampling results for 55X PacBio NA12878 data. (c) Subsampling results for 28X Oxford Nanopore NA12878 data. (d) Subsampling results for the 70x PacBio SKBR3 breast cancer cell line dataset.

25X coverage is needed for paired-end 100-bp short reads (Fig. 4.16a). However, this simulation ignores important factors like the repeat content of the human genome, read coverage biases, mapping bias and sequencing error and thus underestimates the required coverage. To get a more realistic estimation of the coverage required, we called SV from a 55X human NA12787 PacBio and a 28X NA12878 Oxford Nanopore datasets as well as a 69X PacBio SKBR3 dataset. Next, we downsampled these datasets to 5X, 10X, 15X, 20X, and 30X coverage. For all datasets we called SVs using NGMLR and Sniffles varying the minimum number of required supporting reads from 1 to 10. Finally, we computed precision and recall for all 50 callsets by comparing them to the SV calls from the initial full-coverage dataset. We found that requiring only one or two supporting reads led to an unacceptably high number false positives (not shown). Thus, we ignored these call-sets and focused on call-sets with a precision of 80% or higher. Out of these the highest recall for the 15X PacBio datasets was 69.6% and 67.2% for NA12878 and SKBR3 respectively (Fig. 4.16b,d). The difference in recall was largely due to the complexity of the SKBR3 cancer sample, which has several extreme copy number amplifications ($>$ 20-fold). When we increased the coverage to 30X, NGMLR and Sniffles showed 80.0% and 76.6% recall with a precision of 85% for NA12878 and SKBR3, respectively. For the Oxford Nanopore NA12878 dataset, the highest recall was 84.2% for 20X coverage (Fig. 4.16c). Note, that the initial Nanopore dataset was 28X so the results are not directly comparable to the PacBio results.

4.3.6 Runtime and memory usage comparison

We benchmarked the different long read mappers used in this study on a PC equipped with 4 x AMD Opteron 6348 Processors with each having 12 cores and 512 GB RAM in total. Every program was executed with 10 threads to map subsampled NA12878 Nanopore (Jain et al., 2018) and PacBio (Zook et al., 2014) NA12878 data. MECAT was the fastest with 1,236 seconds while NGMLR took 4,788 seconds (1.3 hours) followed by BWA-MEM with 6,133 seconds (1.7 hours). BlasR took 18,518 seconds (5.1 hours). We stopped GraphMap after running for a week without finishing. GraphMap used the most memory (106.9 GB), whereas BWA-

MEM used the least (6.0 Gbytes). NGMLR used 10.2 GB. We also benchmarked Sniffles on the full datasets mapped by NGMLR: Sniffles required 12,102 seconds (3.3 hours) for the 44x PacBio data set, and 7,744 seconds (2.2 hours) for the 28x Nanopore data set. Supplementary Table B.9 shows all results including memory usage for all mappers.

4.4 Conclusion

NGMLR and Sniffles enable an unprecedented view of SVs with long-read sequencing, by outperforming existing tools in terms of both sensitivity and specificity with simulated and real data. In particular, we demonstrated that they can overcome the sensitivity issues reported for short-read callers, which miss 30% (Sudmant, 2015, Jeffares et al., 2017) to 90% (Huddleston, 2017) of SVs. This allowed us to detect many thousands of additional variants beyond what has been reported for large-scale short-read sequencing projects such as the 1000 Genomes Project. Indeed, prototype versions of our methods were recently used to identify the causal pathogenic SV in a patient with multiple neoplasia and cardiac myxomata (Merker et al., 2017). We also used long-read data to identify systematic errors in short-read SV calling, for which the vast majority ($>$ 85%) of identified translocations were false positives caused by mismapped reads. The identification of SVs from long reads is challenging chiefly because of the high underlying error rates. In addition to numerous small indels, we discovered that PacBio introduces larger insertion artifacts at a low but noticeable rate. We control for this artifact by requiring that the size and composition of candidate SVs be consistent across the spanning reads. Within the Oxford Nanopore dataset, we highlighted systematic artifacts in base-calling that generate deletions in low-complexity repeats. Although we fully expect accuracy to improve with improved base-calling, it is currently necessary to exclude small ($<$200bp) SV calls when using Oxford Nanopore sequencing. Beyond sequencing errors, alignment artifacts can lead to false positive SVs. For example, some long-read mappers falsely align reads across a SV without indication of the underlying event. Although Sniffles recognizes the increase in mismatches, NGMLR alignments correct these issues more directly. Finally,

we showed NGMRL and Sniffles to be the only tools that reliably detect nested or complex SVs such as INVDUPs and INVDELs. Several diseases are already known to be associated with these SV types, and we expect that their importance will increase as more are detected. A key remaining barrier to routine analysis of SVs across large numbers of samples is cost. Long-read sequencing is becoming less expensive every year but remains more expensive than short-read sequencing (Goodwin et al., 2016). We addressed this by showing that high accuracy is possible with as little as 15X to 30X coverage for healthy or cancerous human genomes. These requirements will be further reduced as read lengths increase and error rates drop. We expect that these improvements, aided by NGMLR and Sniffles, will usher in a new era of high-quality genome sequences for a broad range of research and clinical applications.

Chapter 5

Summary and Outlook

The goal of this thesis was to develop easy-to-use read mapping tools that map reads efficiently and accurately independent of the number of differences between the reads and the reference genome. We, therefore, set off by discussing the basics of high-throughput next-generation sequencing workflows and summarised the main challenges that need to be addressed by read mapping tools. We found that a wide range of analysis tools exist that are optimised for the more common use-cases like mapping of high accuracy reads to high quality reference genomes followed by SNP and small indel calling. However, with the decreasing cost and increasing availability of next-generation sequencing we saw a need for efficient, accurate and easy-to-use read mapping tools that address challenges that come with the analysis of data from none model organisms, sequencing platforms with higher sequencing error or other types of genetic variants like structural variations.

To address this, in chapter 3 we presented NextGenMap a sequence alignment-based short read mapper. We introduced a *k*-mer index that is optimised for read mapping by enabling fast querying of *k*-mer positions, while being compact enough to fit into main memory on standard desktop computers. Next, we demonstrated how we use a read specific *k*-mer filtering strategy to quickly identify regions in the reference genome with high similarity to a given read. Most importantly, we show this approach to work efficiently independent of the overall number of differences between the reads and the reference (up to 10%). Paired with low-level optimisations like using efficient bit-wise operations e.g., for *k*-mer representation

and extraction as well as using CPU cache efficiently, these two innovations enable NextGenMap to map reads faster and more accurate than state of the art BWT-based aligners. We found NextGenMap to be between 1.1 and 2.3 times faster than the fastest BWT-based mapper. At the same time, NextGenMap maps roughly the same number of reads correctly as BWT-based mappers at low levels of differences between the reads and the reference (1-2%) and up to 46% more at higher levels. Furthermore, NextGenMap is between 11 and 67 times faster than a sequence alignment-based tool that gives similar alignment accuracy at higher polymorphism rates.

Currently, NextGenMap only supports reference genomes smaller than 4GB. This is sufficient for the majority of the currently studied genomes including the human genome. However, with more large plant genome assemblies becoming available it would be beneficial to remove this restriction in the future. NextGenMap's index data-structure can be easily extended to support larger genomes. The index used by NextGenMap comprises two data-structures. An array (GP) used to store the genomic positions of all k-mers and a hash-table (HT) that enables fast querying of GP. To this end, HT uses the k-mer sequence as a key and stores the information where the genomic positions of the respective k-mer are found in GP, as a value. The limiting factor for the reference genome size is that both, the values in HT as well as the elements of GP, are unsigned 32-bit integers. A straightforward approach to resolve this issue is to double the size to 64-bit. However, this would make the index twice as big - even for genomes $<$ 4GB. A more memory efficient strategy would be to split the reference into two 4GB chunks and compute HT_1 and GP_1 for the first part and HT_2 and GP_2 for the second. All genomic position $<$ 4GB would be stored in GP_1 all others in GP_2. Querying HT_1 and GP_1 would be unchanged compared to the current implementation. For GP_2 the only difference would be that all genomic positions would have to be offset by 2^{32} when querying k-mer positions. This approach would double the supported reference size. Querying k-mer position would take slightly longer as it would require two hash-table look-ups instead of one and two read operations if the reference size is $>$ 8GB. However, k-mer position querying is currently not the bottleneck in NextGenMap. Thus, real world runtimes should not be noticeably impacted. Evidently, this strategy can be

extended to use four or more *GP* tables to increase the size of supported reference genomes even further.

The SAM specification defines mapping quality as the probability of the reported mapping position to be wrong (expressed as a Phred quality score). In NextGenMap we currently use the relative difference between the best and the second-best alignment score scaled to a number between 0 and 60 as a proxy for this probability. We have found this approach to approximate mapping quality well for all practical purposes. However, a potential next step would be to use a probabilistic framework to calibrate the mapping qualities and thus mimic the probability of the mapping to be wrong more accurately. This calibration would most likely have to be done separately for different sequencing platforms and might need adjustment depending on how big the evolutionary distance between the sequenced and the reference genome is. Furthermore, no base qualities are taken into account for match and mismatch scores in the scoring function of the sequence alignment algorithms NextGenMap uses. This is a limitation of the MAson library. Extending MAson or switching to another alignment library would be possible and should come at little to no computational overhead when running on a CPU. Due to the limited amount of memory and the additional data transfer overhead required it could however noticeably influence the alignment performance when using a GPU. Furthermore, while easy to apply to platforms with mostly substitution-based sequencing error it is unclear how to extend this approach to indel based sequencing errors as seen in 454 or IonTorrent sequencing. Lastly, NextGenMap was not optimised for detecting split read alignments required for reads that span break points of SVs. A straightforward approach would be to find the primary mapping location as it is implemented now and scan for unaligned bases at the 5' or 3' end of the aligned read. If the number of unaligned bases exceeds a threshold the respective part of the read would be remapped.

However, SV calling from short reads comes with considerable limitations leading to low precision and recall. Thus, to address SV calling, we decided to focus on long read sequencing in chapter 4 instead. We found that existing long read aligners were unable to account for indel based sequencing error and biological variation at the same time. This led to alignment artifacts which significantly lim-

ited SV calling performance. Furthermore, established tools were not optimised for aligning reads that span complex SV and therefore require accurate split read alignments. In addition, no long-read optimised SV caller existed that supported calling of all SV types. To enable researchers to investigate the important role of structural variation in a cost-efficient manner using long reads we introduced NGMLR, an SV aware long read aligner and its companion tool Sniffles, a long-read optimised SV caller. For NGMLR the key innovations were: (1) the introduction of a heuristic implementation of a convex gap cost alignment algorithm that is able to account for sequencing error and true biological SVs while being fast enough to be applied to whole genome datasets. (2) a step wise alignment approach that first finds linear alignments between the reads and the reference and then chooses the non-overlapping subset of linear alignments that taken together result in the highest alignment scores in a SV-aware fashion.

We showed that NGMLR aligns 92% of all reads spanning SVs correctly when using simulated data. Compared to 69% achieved by the second-best long read aligner. NGMLR and Sniffles correctly identified 94% of all simulated SVs with an FDR of 0 % compared to 79% with an FDR of 0.7% for the second-best long read aligner. For comparison, using state of the art short read SV calling tools only recovered 19% of the SVs with an FDR of 0.2% when applying the same evaluation criteria. When looking at a PacBio trio dataset NGMLR and Sniffles detected $\sim$ 94% of all paternal and maternal homozygous SVs in the son as well. Using short reads four times more SVs were missed. Finally, we showed that highly accurate SV calling is possible using 15 - 30 times coverage when using long reads making our approach generally applicable and cost effective.

Two notably limitations of NGMLR are: (1) it can only map to one linear reference genome and (2) as it was optimised for aligning raw long sequencing reads it currently does not support whole genome alignments. With the increasing availability of long read sequencing, more and more assemblies of human genomes as well as growing catalogues of common SVs will be available. Thus, adding support to NGMLR for aligning full genome assemblies as well as aligning to a reference genome graph that contains information of common SVs on top of the reference sequence might be of general interest. A more practical limitation is that NGMLR

currently implements stringent read length filters. Since short single molecules sequencing reads are generally not useful for SV calling NGMLR discards reads shorter than a given length ($<$ 1 kb by default). While not affecting overall SV calling performance this reduces the percentage of aligned reads. With long read sequencing platforms becoming more accurate and thus being applied to small variant calling as well, revisiting these thresholds could increase the overall applicability of NGMLR.

Similar to the development seen for short read data we expect more tools becoming available for long-read alignment and SV calling in the future. A robust approach for benchmarking of these tools will be required. In chapter 4 we introduced an approach to evaluate read alignments and SV calls from simulated data and trio datasets. Consolidating these evaluations in an easy-to-use framework to allow for automatic and comprehensive evaluation of SV calling tools could be of great benefit to the SV calling community.

Bibliography

Alberts B, Johnson A, and Lewis J (2002): Molecular Biology of the Cell. 4th edition, New York: Garland Science, 4th edition, URL https://www.ncbi.nlm.nih.gov/books/NBK21054/.

Alkan, C., B. P. Coe, and E. E. Eichler (2011): "Genome structural variation discovery and genotyping," Nature Reviews Genetics, 12, 363–376, URL http://www.ncbi.nlm.nih.gov/pubmed/21358748http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4108431http://www.nature.com/doifinder/10.1038/nrg2958.

Bębenek, A. and I. Ziuzia-Graczyk (2018): "Fidelity of DNA replicationa matter of proofreading," .

Beri, S., M. C. Bonaglia, and R. Giorda (2013): "Low-copy repeats at the human VIPR2 gene predispose to recurrent and nonrecurrent rearrangements." European journal of human genetics : EJHG, 21, 757–61, URL http://www.nature.com/doifinder/10.1038/ejhg.2012.235http://www.ncbi.nlm.nih.gov/pubmed/23073313http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3722940.

Blom, J., T. Jakobi, D. Doppmeier, S. Jaenicke, J. Kalinowski, J. Stoye, and A. Goesmann (2011): "Exact and complete short-read alignment to microbial genomes using graphics processing unit programming," Bioinformatics, 27, 1351–1358.

Canzar, S. and S. L. Salzberg (2015): "Short Read Mapping: An Algorithmic Tour," Proceedings of the IEEE, 1–23, URL http://ieeexplore.ieee.org/document/7244195/.

Carvalho, C. M. B. and J. R. Lupski (2016): "Mechanisms underlying structural variant formation in genomic disorders," Nature Reviews Genetics, 17, 224–238, URL http://www.ncbi.nlm.nih.gov/pubmed/26924765http://www.

pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4827625http://www.nature.com/doifinder/10.1038/nrg.2015.25.

Carvalho, C. M. B., M. B. Ramocki, D. Pehlivan, L. M. Franco, C. Gonzaga-Jauregui, P. Fang, A. McCall, E. K. Pivnick, S. Hines-Dowell, L. H. Seaver, L. Friehling, S. Lee, R. Smith, D. Del Gaudio, M. Withers, P. Liu, S. W. Cheung, J. W. Belmont, H. Y. Zoghbi, P. J. Hastings, and J. R. Lupski (2011): "Inverted genomic segments and complex triplication rearrangements are mediated by inverted repeats in the human genome." Nature genetics, 43, 1074–81, URL http://www.nature.com/doifinder/10.1038/ng.944http://www.ncbi.nlm.nih.gov/pubmed/21964572http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3235474.

Cechova, M. (2021): "Probably correct: Rescuing repeats with short and long reads," .

Chaisson, M. J. (2015): "Resolving the complexity of the human genome using single-molecule sequencing," Nature, 517, URL https://doi.org/10.1038/nature13907.

Chaisson, M. J. and G. Tesler (2012): "Mapping single molecule sequencing reads using basic local alignment with successive refinement (BLASR): application and theory," BMC Bioinformatics, 13, 238, URL http://www.ncbi.nlm.nih.gov/pubmed/22988817http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3572422http://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-13-238.

Chen, X., O. Schulz-Trieglaff, R. Shaw, B. Barnes, F. Schlesinger, M. Källberg, A. J. Cox, S. Kruglyak, and C. T. Saunders (2016): "Manta: rapid detection of structural variants and indels for germline and cancer sequencing applications." Bioinformatics (Oxford, England), 32, 1220–2, URL https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/

btv710http://www.ncbi.nlm.nih.gov/pubmed/26647377.

Chin, C.-S., P. Peluso, F. J. Sedlazeck, M. Nattestad, G. T. Concepcion, A. Clum, C. Dunn, R. O'Malley, R. Figueroa-Balderas, A. Morales-Cruz, G. R. Cramer, M. Delledonne, C. Luo, J. R. Ecker, D. Cantu, D. R. Rank, and M. C. Schatz (2016): "Phased diploid genome assembly with single-molecule real-time sequencing," Nature Methods, 13, 1050–1054, URL http://www.ncbi.nlm.nih.gov/pubmed/27749838http://www.nature.com/doifinder/10.1038/nmeth.4035.

Chong, J. X., K. J. Buckingham, S. N. Jhangiani, C. Boehm, N. Sobreira, J. D. Smith, T. M. Harrell, M. J. McMillin, W. Wiszniewski, T. Gambin, Z. H. Coban Akdemir, K. Doheny, A. F. Scott, D. Avramopoulos, A. Chakravarti, J. Hoover-Fong, D. Mathews, P. D. Witmer, H. Ling, K. Hetrick, L. Watkins, K. E. Patterson, F. Reinier, E. Blue, D. Muzny, M. Kircher, K. Bilguvar, F. López-Giráldez, V. R. Sutton, H. K. Tabor, S. M. Leal, M. Gunel, S. Mane, R. A. Gibbs, E. Boerwinkle, A. Hamosh, J. Shendure, J. R. Lupski, R. P. Lifton, D. Valle, D. A. Nickerson, and M. J. Bamshad (2015): "The Genetic Basis of Mendelian Phenotypes: Discoveries, Challenges, and Opportunities," .

Craig Venter, J., M. D. Adams, E. W. Myers, P. W. Li, R. J. Mural, G. G. Sutton, H. O. Smith, M. Yandell, C. A. Evans, R. A. Holt, J. D. Gocayne, P. Amanatides, R. M. Ballew, D. H. Huson, J. R. Wortman, Q. Zhang, C. D. Kodira, X. H. Zheng, L. Chen, M. Skupski, G. Subramanian, P. D. Thomas, J. Zhang, G. L. Gabor Miklos, C. Nelson, S. Broder, A. G. Clark, J. Nadeau, V. A. McKusick, N. Zinder, A. J. Levine, R. J. Roberts, M. Simon, C. Slayman, M. Hunkapiller, R. Bolanos, A. Delcher, I. Dew, D. Fasulo, M. Flanigan, L. Florea, A. Halpern, S. Hannenhalli, S. Kravitz, S. Levy, C. Mobarry, K. Reinert, K. Remington, J. Abu-Threideh, E. Beasley, K. Biddick, V. Bonazzi, R. Brandon, M. Cargill, I. Chandramouliswaran, R. Charlab, K. Chaturvedi, Z. Deng, V. di Francesco, P. Dunn, K. Eilbeck, C. Evangelista, A. E. Gabrielian, W. Gan, W. Ge, F. Gong, Z. Gu, P. Guan, T. J. Heiman, M. E. Higgins, R. R. Ji, Z. Ke, K. A. Ketchum, Z. Lai, Y. Lei, Z. Li, J. Li, Y. Liang, X. Lin, F. Lu, G. V. Merkulov, N. Mil-

shina, H. M. Moore, A. K. Naik, V. A. Narayan, B. Neelam, D. Nusskern, D. B. Rusch, S. Salzberg, W. Shao, B. Shue, J. Sun, Z. Yuan Wang, A. Wang, X. Wang, J. Wang, M. H. Wei, R. Wides, C. Xiao, C. Yan, A. Yao, J. Ye, M. Zhan, W. Zhang, H. Zhang, Q. Zhao, L. Zheng, F. Zhong, W. Zhong, S. C. Zhu, S. Zhao, D. Gilbert, S. Baumhueter, G. Spier, C. Carter, A. Cravchik, T. Woodage, F. Ali, H. An, A. Awe, D. Baldwin, H. Baden, M. Barnstead, I. Barrow, K. Beeson, D. Busam, A. Carver, A. Center, M. Lai Cheng, L. Curry, S. Danaher, L. Davenport, R. Desilets, S. Dietz, K. Dodson, L. Doup, S. Ferriera, N. Garg, A. Gluecksmann, B. Hart, J. Haynes, C. Haynes, C. Heiner, S. Hladun, D. Hostin, J. Houck, T. Howland, C. Ibegwam, J. Johnson, F. Kalush, L. Kline, S. Koduru, A. Love, F. Mann, D. May, S. McCawley, T. McIntosh, I. McMullen, M. Moy, L. Moy, B. Murphy, K. Nelson, C. Pfannkoch, E. Pratts, V. Puri, H. Qureshi, M. Reardon, R. Rodriguez, Y. H. Rogers, D. Romblad, B. Ruhfel, R. Scott, C. Sitter, M. Smallwood, E. Stewart, R. Strong, E. Suh, R. Thomas, N. Ni Tint, S. Tse, C. Vech, G. Wang, J. Wetter, S. Williams, M. Williams, S. Windsor, E. Winn-Deen, K. Wolfe, J. Zaveri, K. Zaveri, J. F. Abril, R. Guigo, M. J. Campbell, K. V. Sjolander, B. Karlak, A. Kejariwal, H. Mi, B. Lazareva, T. Hatton, A. Narechania, K. Diemer, A. Muruganujan, N. Guo, S. Sato, V. Bafna, S. Istrail, R. Lippert, R. Schwartz, B. Walenz, S. Yooseph, D. Allen, A. Basu, J. Baxendale, L. Blick, M. Caminha, J. Carnes-Stine, P. Caulk, Y. H. Chiang, M. Coyne, C. Dahlke, A. Deslattes Mays, M. Dombroski, M. Donnelly, D. Ely, S. Esparham, C. Fosler, H. Gire, S. Glanowski, K. Glasser, A. Glodek, M. Gorokhov, K. Graham, B. Gropman, M. Harris, J. Heil, S. Henderson, J. Hoover, D. Jennings, C. Jordan, J. Jordan, J. Kasha, L. Kagan, C. Kraft, A. Levitsky, M. Lewis, X. Liu, J. Lopez, D. Ma, W. Majoros, J. McDaniel, S. Murphy, M. Newman, T. Nguyen, N. Nguyen, M. Nodell, S. Pan, J. Peck, M. Peterson, W. Rowe, R. Sanders, J. Scott, M. Simpson, T. Smith, A. Sprague, T. Stockwell, R. Turner, E. Venter, M. Wang, M. Wen, D. Wu, M. Wu, A. Xia, A. Zandieh, and X. Zhu (2001): "The sequence of the human genome," Science, 291.

Crick, F. (1970): "Central dogma of molecular biology," Nature, 227.

de Koning, A. P., W. Gu, T. A. Castoe, M. A. Batzer, and D. D. Pollock (2011):

"Repetitive elements may comprise over Two-Thirds of the human genome," PLoS Genetics, 7.

Dennenmoser, S. (2017): "Copy number increases of transposable elements and protein-coding genes in an invasive fish of hybrid origin," Mol. Ecol., 26, URL https://doi.org/10.1111/mec.14134.

Dinh, H., M. Dubin, F. Sedlazeck, N. Lettner, O. Mittelsten Scheid, and A. von Haeseler (2012): "Advanced methylome analysis after Bisulfite deep sequencing: An example in Arabidopsis," PLoS ONE, 7.

Eberle, M. A., E. Fritzilas, P. Krusche, M. Källberg, B. L. Moore, M. A. Bekritsky, Z. Iqbal, H.-Y. Chuang, S. J. Humphray, A. L. Halpern, S. Kruglyak, E. H. Margulies, G. McVean, and D. R. Bentley (2017): "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, 157–164, URL http://genome.cshlp.org/lookup/doi/10.1101/gr.210500.116http://www.ncbi.nlm.nih.gov/pubmed/27903644http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC5204340.

English, A. C., W. J. Salerno, and J. G. Reid (2014): "PBHoney: identifying genomic variants via long-read discordance and interrupted mapping." BMC bioinformatics, 15, 180, URL http://www.ncbi.nlm.nih.gov/pubmed/24915764http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4082283.

Ferragina, P. and G. Manzini (2005): "Indexing compressed text," Journal of the ACM, 52.

Fonseca, N., J. Rung, A. Brazma, and J. Marioni (2012): "Tools for mapping high-throughput sequencing data," Bioinformatics, 28, 3169–3177.

Gibbs, R. A., E. Boerwinkle, H. Doddapaneni, Y. Han, V. Korchina, C. Kovar, S. Lee, D. Muzny, J. G. Reid, Y. Zhu, J. Wang, Y. Chang, Q. Feng, X. Fang,

X. Guo, M. Jian, H. Jiang, X. Jin, T. Lan, G. Li, J. Li, Y. Li, S. Liu, X. Liu, Y. Lu, X. Ma, M. Tang, B. Wang, G. Wang, H. Wu, R. Wu, X. Xu, Y. Yin, D. Zhang, W. Zhang, J. Zhao, M. Zhao, X. Zheng, E. S. Lander, D. M. Altshuler, S. B. Gabriel, N. Gupta, N. Gharani, L. H. Toji, N. P. Gerry, A. M. Resch, P. Flicek, J. Barker, L. Clarke, L. Gil, S. E. Hunt, G. Kelman, E. Kulesha, R. Leinonen, W. M. McLaren, R. Radhakrishnan, A. Roa, D. Smirnov, R. E. Smith, I. Streeter, A. Thormann, I. Toneva, B. Vaughan, X. Zheng-Bradley, D. R. Bentley, R. Grocock, S. Humphray, T. James, Z. Kingsbury, H. Lehrach, R. Sudbrak, M. W. Albrecht, V. S. Amstislavskiy, T. A. Borodina, M. Lienhard, F. Mertes, M. Sultan, B. Timmermann, M.-L. Yaspo, E. R. Mardis, R. K. Wilson, L. Fulton, R. Fulton, S. T. Sherry, V. Ananiev, Z. Belaia, D. Beloslyudtsev, N. Bouk, C. Chen, D. Church, R. Cohen, C. Cook, J. Garner, T. Hefferon, M. Kimelman, C. Liu, J. Lopez, P. Meric, C. OSullivan, Y. Ostapchuk, L. Phan, S. Ponomarov, V. Schneider, E. Shekhtman, K. Sirotkin, D. Slotta, H. Zhang, G. A. McVean, R. M. Durbin, S. Balasubramaniam, J. Burton, P. Danecek, T. M. Keane, A. Kolb-Kokocinski, S. McCarthy, J. Stalker, M. Quail, J. P. Schmidt, C. J. Davies, J. Gollub, T. Webster, B. Wong, Y. Zhan, A. Auton, C. L. Campbell, Y. Kong, A. Marcketta, R. A. Gibbs, F. Yu, L. Antunes, M. Bainbridge, D. Muzny, A. Sabo, Z. Huang, J. Wang, L. J. M. Coin, L. Fang, X. Guo, X. Jin, G. Li, Q. Li, Y. Li, Z. Li, H. Lin, B. Liu, R. Luo, H. Shao, Y. Xie, C. Ye, C. Yu, F. Zhang, H. Zheng, H. Zhu, C. Alkan, E. Dal, F. Kahveci, G. T. Marth, E. P. Garrison, D. Kural, W.-P. Lee, W. Fung Leong, M. Stromberg, A. N. Ward, J. Wu, M. Zhang, M. J. Daly, M. A. DePristo, R. E. Handsaker, D. M. Altshuler, E. Banks, G. Bhatia, G. del Angel, S. B. Gabriel, G. Genovese, N. Gupta, H. Li, S. Kashin, E. S. Lander, S. A. McCarroll, J. C. Nemesh, R. E. Poplin, S. C. Yoon, J. Lihm, V. Makarov, A. G. Clark, S. Gottipati, A. Keinan, J. L. Rodriguez-Flores, J. O. Korbel, T. Rausch, M. H. Fritz, A. M. Stütz, P. Flicek, K. Beal, L. Clarke, A. Datta, J. Herrero, W. M. McLaren, G. R. S. Ritchie, R. E. Smith, D. Zerbino, X. Zheng-Bradley, P. C. Sabeti, I. Shlyakhter, S. F. Schaffner, J. Vitti, D. N. Cooper, E. V. Ball, P. D. Stenson, D. R. Bentley, B. Barnes, M. Bauer, R. Keira Cheetham, A. Cox, M. Eberle, S. Humphray, S. Kahn, L. Murray, J. Peden, R. Shaw, E. E. Kenny, M. A. Batzer, M. K. Konkel, J. A. Walker, D. G. MacArthur, M. Lek, R. Sud-

brak, V. S. Amstislavskiy, R. Herwig, E. R. Mardis, L. Ding, D. C. Koboldt, D. Larson, K. Ye, S. Gravel, A. Swaroop, E. Chew, T. Lappalainen, Y. Erlich, M. Gymrek, T. Frederick Willems, J. T. Simpson, M. D. Shriver, J. A. Rosenfeld, C. D. Bustamante, S. B. Montgomery, F. M. De La Vega, J. K. Byrnes, A. W. Carroll, M. K. DeGorter, P. Lacroute, B. K. Maples, A. R. Martin, A. Moreno-Estrada, S. S. Shringarpure, F. Zakharia, E. Halperin, Y. Baran, C. Lee, E. Cerveira, J. Hwang, A. Malhotra, D. Plewczynski, K. Radew, M. Romanovitch, C. Zhang, F. C. L. Hyland, D. W. Craig, A. Christoforides, N. Homer, T. Izatt, A. A. Kurdoglu, S. A. Sinari, K. Squire, S. T. Sherry, C. Xiao, J. Sebat, D. Antaki, M. Gujral, A. Noor, K. Ye, E. G. Burchard, R. D. Hernandez, C. R. Gignoux, D. Haussler, S. J. Katzman, W. James Kent, B. Howie, A. Ruiz-Linares, E. T. Dermitzakis, S. E. Devine, G. R. Abecasis, H. Min Kang, J. M. Kidd, T. Blackwell, S. Caron, W. Chen, S. Emery, L. Fritsche, C. Fuchsberger, G. Jun, B. Li, R. Lyons, C. Scheller, C. Sidore, S. Song, E. Sliwerska, D. Taliun, A. Tan, R. Welch, M. Kate Wing, X. Zhan, P. Awadalla, A. Hodgkinson, Y. Li, X. Shi, A. Quitadamo, G. Lunter, G. A. McVean, J. L. Marchini, S. Myers, C. Churchhouse, O. Delaneau, A. Gupta-Hinch, W. Kretzschmar, Z. Iqbal, I. Mathieson, A. Menelaou, A. Rimmer, D. K. Xifara, T. K. Oleksyk, Y. Fu, X. Liu, M. Xiong, L. Jorde, D. Witherspoon, J. Xing, E. E. Eichler, B. L. Browning, S. R. Browning, F. Hormozdiari, P. H. Sudmant, E. Khurana, R. M. Durbin, M. E. Hurles, C. Tyler-Smith, C. A. Albers, Q. Ayub, S. Balasubramaniam, Y. Chen, V. Colonna, P. Danecek, L. Jostins, T. M. Keane, S. McCarthy, K. Walter, Y. Xue, M. B. Gerstein, A. Abyzov, S. Balasubramanian, J. Chen, D. Clarke, Y. Fu, A. O. Harmanci, M. Jin, D. Lee, J. Liu, X. Jasmine Mu, J. Zhang, Y. Zhang, Y. Li, R. Luo, H. Zhu, C. Alkan, E. Dal, F. Kahveci, G. T. Marth, E. P. Garrison, D. Kural, W.-P. Lee, A. N. Ward, J. Wu, M. Zhang, S. A. McCarroll, R. E. Handsaker, D. M. Altshuler, E. Banks, G. del Angel, G. Genovese, C. Hartl, H. Li, S. Kashin, J. C. Nemesh, K. Shakir, S. C. Yoon, J. Lihm, V. Makarov, J. Degenhardt, J. O. Korbel, M. H. Fritz, S. Meiers, B. Raeder, T. Rausch, A. M. Stütz, P. Flicek, F. Paolo Casale, L. Clarke, R. E. Smith, O. Stegle, X. Zheng-Bradley, D. R. Bentley, B. Barnes, R. Keira Cheetham, M. Eberle, S. Humphray, S. Kahn, L. Murray, R. Shaw, E.-W. Lameijer, M. A. Batzer, M. K. Konkel, J. A.

Walker, L. Ding, I. Hall, K. Ye, P. Lacroute, C. Lee, E. Cerveira, A. Malhotra, J. Hwang, D. Plewczynski, K. Radew, M. Romanovitch, C. Zhang, D. W. Craig, N. Homer, D. Church, C. Xiao, J. Sebat, D. Antaki, V. Bafna, J. Michaelson, K. Ye, S. E. Devine, E. J. Gardner, G. R. Abecasis, J. M. Kidd, R. E. Mills, G. Dayama, S. Emery, G. Jun, X. Shi, A. Quitadamo, G. Lunter, G. A. McVean, K. Chen, X. Fan, Z. Chong, T. Chen, D. Witherspoon, J. Xing, E. E. Eichler, M. J. Chaisson, F. Hormozdiari, J. Huddleston, M. Malig, B. J. Nelson, P. H. Sudmant, N. F. Parrish, E. Khurana, M. E. Hurles, B. Blackburne, S. J. Lindsay, Z. Ning, K. Walter, Y. Zhang, M. B. Gerstein, A. Abyzov, J. Chen, D. Clarke, H. Lam, X. Jasmine Mu, C. Sisu, J. Zhang, Y. Zhang, R. A. Gibbs, F. Yu, M. Bainbridge, D. Challis, U. S. Evani, C. Kovar, J. Lu, D. Muzny, U. Nagaswamy, J. G. Reid, A. Sabo, J. Yu, X. Guo, W. Li, Y. Li, R. Wu, G. T. Marth, E. P. Garrison, W. Fung Leong, A. N. Ward, G. del Angel, M. A. DePristo, S. B. Gabriel, N. Gupta, C. Hartl, R. E. Poplin, A. G. Clark, J. L. Rodriguez-Flores, P. Flicek, L. Clarke, R. E. Smith, X. Zheng-Bradley, D. G. MacArthur, E. R. Mardis, R. Fulton, D. C. Koboldt, S. Gravel, C. D. Bustamante, D. W. Craig, A. Christoforides, N. Homer, T. Izatt, S. T. Sherry, C. Xiao, E. T. Dermitzakis, G. R. Abecasis, H. Min Kang, G. A. McVean, M. B. Gerstein, S. Balasubramanian, L. Habegger, H. Yu, P. Flicek, L. Clarke, F. Cunningham, I. Dunham, D. Zerbino, X. Zheng-Bradley, K. Lage, J. Berg Jespersen, H. Horn, S. B. Montgomery, M. K. DeGorter, E. Khurana, C. Tyler-Smith, Y. Chen, V. Colonna, Y. Xue, M. B. Gerstein, S. Balasubramanian, Y. Fu, D. Kim, A. Auton, A. Marcketta, R. Desalle, A. Narechania, M. A. Wilson Sayres, E. P. Garrison, R. E. Handsaker, S. Kashin, S. A. McCarroll, J. L. Rodriguez-Flores, P. Flicek, L. Clarke, X. Zheng-Bradley, Y. Erlich, M. Gymrek, T. Frederick Willems, C. D. Bustamante, F. L. Mendez, G. David Poznik, P. A. Underhill, C. Lee, E. Cerveira, A. Malhotra, M. Romanovitch, C. Zhang, G. R. Abecasis, L. Coin, H. Shao, D. Mittelman, C. Tyler-Smith, Q. Ayub, R. Banerjee, M. Cerezo, Y. Chen, T. W. Fitzgerald, S. Louzada, A. Massaia, S. McCarthy, G. R. Ritchie, Y. Xue, F. Yang, R. A. Gibbs, C. Kovar, D. Kalra, W. Hale, D. Muzny, J. G. Reid, J. Wang, X. Dan, X. Guo, G. Li, Y. Li, C. Ye, X. Zheng, D. M. Altshuler, P. Flicek, L. Clarke, X. Zheng-Bradley, D. R. Bentley, A. Cox, S. Humphray, S. Kahn,

R. Sudbrak, M. W. Albrecht, M. Lienhard, D. Larson, D. W. Craig, T. Izatt, A. A. Kurdoglu, S. T. Sherry, C. Xiao, D. Haussler, G. R. Abecasis, G. A. McVean, R. M. Durbin, S. Balasubramaniam, T. M. Keane, S. McCarthy, J. Stalker, W. Bodmer, G. Bedoya, A. Ruiz-Linares, Z. Cai, Y. Gao, J. Chu, L. Peltonen, A. Garcia-Montero, A. Orfao, J. Dutil, J. C. Martinez-Cruzado, T. K. Oleksyk, K. C. Barnes, R. A. Mathias, A. Hennis, H. Watson, C. McKenzie, F. Qadri, R. LaRocque, P. C. Sabeti, J. Zhu, X. Deng, P. C. Sabeti, D. Asogun, O. Folarin, C. Happi, O. Omoniwa, M. Stremlau, R. Tariyal, M. Jallow, F. Sisay Joof, T. Corrah, K. Rockett, D. Kwiatkowski, J. Kooner, T. Tnh Hiê‘n, S. J. Dunstan, N. Thuy Hang, R. Fonnie, R. Garry, L. Kanneh, L. Moses, P. C. Sabeti, J. Schieffelin, D. S. Grant, C. Gallo, G. Poletti, D. Saleheen, and A. Rasheed (2015): "A global reference for human genetic variation," Nature, 526, 68–74, URL http://www.nature.com/articles/nature15393.

Glenn, T. C. (2011): "Field guide to next-generation DNA sequencers." Molecular ecology resources, 759–769, URL http://www.ncbi.nlm.nih.gov/pubmed/21592312.

Goodwin, S., J. D. McPherson, and W. R. McCombie (2016): "Coming of age: ten years of next-generation sequencing technologies," Nature Reviews Genetics, 17, 333–351, URL http://www.ncbi.nlm.nih.gov/pubmed/27184599http://www.nature.com/doifinder/10.1038/nrg.2016.49.

Gusfield, D. (1997): Algorithms on strings, trees, and sequences: computer science and computational bio New York, NY, USA: Cambridge University Press.

Hartigan, J. A. and P. M. Hartigan (1985): "The Dip Test of Unimodality," Ann. Statist., 13, 70–84, URL https://doi.org/10.1214/aos/1176346577.

Hedges, D. J., K. L. Hamilton-Nelson, S. J. Sacharow, L. Nations, G. W. Beecham, Z. M. Kozhekbaeva, B. L. Butler, H. N. Cukier, P. L. Whitehead, D. Ma, J. M. Jaworski, L. Nathanson, J. M. Lee, S. L. Hauser, J. R. Oksenberg, M. L. Cuccaro, J. L. Haines, J. R. Gilbert, and M. A. Pericak-Vance (2012): "Evidence of novel

fine-scale structural variation at autism spectrum disorder candidate loci." Molecular autism, 3, 2, URL http://molecularautism.biomedcentral.com/articles/10.1186/2040-2392-3-2http://www.ncbi.nlm.nih.gov/pubmed/22472195http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3352055.

Holtgrewe, M. (2010): "Mason a read simulator for second generation sequencing data." Technical Report TR-B-10-06, Institut für Mathematik und Informatik, Freie Universität Berlin.

Huddleston, J. (2017): "Discovery and genotyping of structural variation from long-read haploid genome sequence data," Genome Res., 27, URL https://doi.org/10.1101/gr.214007.116.

Imprialou, M., A. Kahles, J. G. Steffen, E. J. Osborne, X. Gan, J. Lempe, A. Bhomra, E. Belfield, A. Visscher, R. Greenhalgh, N. P. Harberd, R. Goram, J. Hein, A. Robert-Seilaniantz, J. Jones, O. Stegle, P. Kover, M. Tsiantis, M. Nordborg, G. Rätsch, R. M. Clark, and R. Mott (2017): "Genomic Rearrangements in Arabidopsis Considered as Quantitative Traits." Genetics, 205, 1425–1441, URL http://www.genetics.org/lookup/doi/10.1534/genetics.116.192823http://www.ncbi.nlm.nih.gov/pubmed/28179367http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC5378104.

Jain, M., S. Koren, K. H. Miga, J. Quick, A. C. Rand, T. A. Sasani, J. R. Tyson, A. D. Beggs, A. T. Dilthey, I. T. Fiddes, S. Malla, H. Marriott, T. Nieto, J. O'Grady, H. E. Olsen, B. S. Pedersen, A. Rhie, H. Richardson, A. R. Quinlan, T. P. Snutch, L. Tee, B. Paten, A. M. Phillippy, J. T. Simpson, N. J. Loman, and M. Loose (2018): "Nanopore sequencing and assembly of a human genome with ultra-long reads," Nature Biotechnology, 36, 338–345.

Jeffares, D. C., C. Jolly, M. Hoti, D. Speed, L. Shaw, C. Rallis, F. Balloux, C. Dessimoz, J. Bähler, and F. J. Sedlazeck (2017): "Transient structural variations have strong effects on quantitative traits and repro-

ductive isolation in fission yeast." Nature communications, 8, 14061, URL http://www.nature.com/doifinder/10.1038/ncomms14061http://www.ncbi.nlm.nih.gov/pubmed/28117401http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC5286201.

Kadalayil, L., S. Rafiq, M. J. J. Rose-Zerilli, R. J. Pengelly, H. Parker, D. Oscier, J. C. Strefford, W. J. Tapper, J. Gibson, S. Ennis, and A. Collins (2015): "Exome sequence read depth methods for identifying copy number changes." Briefings in bioinformatics, 16, 380–92, URL https://academic.oup.com/bib/article-lookup/doi/10.1093/bib/bbu027http://www.ncbi.nlm.nih.gov/pubmed/25169955.

Kiełbasa, S. M., R. Wan, K. Sato, P. Horton, and M. C. Frith (2011): "Adaptive seeds tame genomic sequence comparison," Genome Res., 21, URL https://doi.org/10.1101/gr.113985.110.

Knuth, D. E. (1997): Art of Computer Programming, Volume 2: Seminumerical Algorithms, Addison-Wesley Professional, third edition, URL http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0201896842.

Koboldt, D. C. (2020): "Best practices for variant calling in clinical sequencing," .

Lander, E. S., L. M. Linton, B. Birren, C. Nusbaum, M. C. Zody, J. Baldwin, K. Devon, K. Dewar, M. Doyle, W. Fitzhugh, R. Funke, D. Gage, K. Harris, A. Heaford, J. Howland, L. Kann, J. Lehoczky, R. Levine, P. McEwan, K. McKernan, J. Meldrim, J. P. Mesirov, C. Miranda, W. Morris, J. Naylor, C. Raymond, M. Rosetti, R. Santos, A. Sheridan, C. Sougnez, N. Stange-Thomann, N. Stojanovic, A. Subramanian, D. Wyman, J. Rogers, J. Sulston, R. Ainscough, S. Beck, D. Bentley, J. Burton, C. Clee, N. Carter, A. Coulson, R. Deadman, P. Deloukas, A. Dunham, I. Dunham, R. Durbin, L. French, D. Grafham, S. Gregory, T. Hubbard, S. Humphray, A. Hunt, M. Jones, C. Lloyd, A. McMurray, L. Matthews, S. Mercer, S. Milne, J. C. Mullikin, A. Mungall, R. Plumb, M. Ross, R. Shownkeen, S. Sims, R. H. Waterston, R. K. Wilson, L. W. Hillier, J. D.

McPherson, M. A. Marra, E. R. Mardis, L. A. Fulton, A. T. Chinwalla, K. H. Pepin, W. R. Gish, S. L. Chissoe, M. C. Wendl, K. D. Delehaunty, T. L. Miner, A. Delehaunty, J. B. Kramer, L. L. Cook, R. S. Fulton, D. L. Johnson, P. J. Minx, S. W. Clifton, T. Hawkins, E. Branscomb, P. Predki, P. Richardson, S. Wenning, T. Slezak, N. Doggett, J. F. Cheng, A. Olsen, S. Lucas, C. Elkin, E. Uberbacher, M. Frazier, R. A. Gibbs, D. M. Muzny, S. E. Scherer, J. B. Bouck, E. J. Sodergren, K. C. Worley, C. M. Rives, J. H. Gorrell, M. L. Metzker, S. L. Naylor, R. S. Kucherlapati, D. L. Nelson, G. M. Weinstock, Y. Sakaki, A. Fujiyama, M. Hattori, T. Yada, A. Toyoda, T. Itoh, C. Kawagoe, H. Watanabe, Y. Totoki, T. Taylor, J. Weissenbach, R. Heilig, W. Saurin, F. Artiguenave, P. Brottier, T. Bruls, E. Pelletier, C. Robert, P. Wincker, A. Rosenthal, M. Platzer, G. Nyakatura, S. Taudien, A. Rump, D. R. Smith, L. Doucette-Stamm, M. Rubenfield, K. Weinstock, M. L. Hong, J. Dubois, H. Yang, J. Yu, J. Wang, G. Huang, J. Gu, L. Hood, L. Rowen, A. Madan, S. Qin, R. W. Davis, N. A. Federspiel, A. P. Abola, M. J. Proctor, B. A. Roe, F. Chen, H. Pan, J. Ramser, H. Lehrach, R. Reinhardt, W. R. McCombie, M. De La Bastide, N. Dedhia, H. Blöcker, K. Hornischer, G. Nordsiek, R. Agarwala, L. Aravind, J. A. Bailey, A. Bateman, S. Batzoglou, E. Birney, P. Bork, D. G. Brown, C. B. Burge, L. Cerutti, H. C. Chen, D. Church, M. Clamp, R. R. Copley, T. Doerks, S. R. Eddy, E. E. Eichler, T. S. Furey, J. Galagan, J. G. Gilbert, C. Harmon, Y. Hayashizaki, D. Haussler, H. Hermjakob, K. Hokamp, W. Jang, L. S. Johnson, T. A. Jones, S. Kasif, A. Kaspryzk, S. Kennedy, W. J. Kent, P. Kitts, E. V. Koonin, I. Korf, D. Kulp, D. Lancet, T. M. Lowe, A. McLysaght, T. Mikkelsen, J. V. Moran, N. Mulder, V. J. Pollara, C. P. Ponting, G. Schuler, J. Schultz, G. Slater, A. F. Smit, E. Stupka, J. Szustakowki, D. Thierry-Mieg, J. Thierry-Mieg, L. Wagner, J. Wallis, R. Wheeler, A. Williams, Y. I. Wolf, K. H. Wolfe, S. P. Yang, R. F. Yeh, F. Collins, M. S. Guyer, J. Peterson, A. Felsenfeld, K. A. Wetterstrand, R. M. Myers, J. Schmutz, M. Dickson, J. Grimwood, D. R. Cox, M. V. Olson, R. Kaul, C. Raymond, N. Shimizu, K. Kawasaki, S. Minoshima, G. A. Evans, M. Athanasiou, R. Schultz, A. Patrinos, and M. J. Morgan (2001): "Initial sequencing and analysis of the human genome," Nature, 409.

Langmead, B. and S. Salzberg (2012): "Fast gapped-read alignment with Bowtie 2," Nature Methods, 9, 357–359.

Langmead, B., C. Trapnell, M. Pop, and S. Salzberg (2009): "Ultrafast and memory-efficient alignment of short {DNA} sequences to the human genome," Genome Biology, 10, R25+, URL http://dx.doi.org/10.1186/gb-2009-10-3-r25.

Layer, R. M., C. Chiang, A. R. Quinlan, and I. M. Hall (2014): "LUMPY: a probabilistic framework for structural variant discovery." Genome biology, 15, R84, URL http://genomebiology.biomedcentral.com/articles/10.1186/gb-2014-15-6-r84http://www.ncbi.nlm.nih.gov/pubmed/24970577http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4197822.

Li, H. (2013): "Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM," .

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

Li, H. and R. Durbin (2010): "Fast and accurate long-read alignment with Burrows-Wheeler transform," Bioinformatics, 26, 589–595.

Li, H., B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis, R. Durbin, and 1000 Genome Project Data Processing Subgroup (2009): "The Sequence Alignment/Map format and SAMtools," Bioinformatics, 25, 2078–2079, URL http://www.ncbi.nlm.nih.gov/pubmed/19505943http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC2723002https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btp352.

Lin, M., S. Whitmire, J. Chen, A. Farrel, X. Shi, and J. T. Guo (2017): "Effects of short indels on protein structure and function in human genomes," Scientific

Reports, 7.

Liu, C.-M., T. Wong, E. Wu, R. Luo, S.-M. Yiu, Y. Li, B. Wang, C. Yu, X. Chu, K. Zhao, R. Li, and T.-W. Lam (2012): "SOAP3: Ultra-fast GPU-based parallel alignment tool for short reads," Bioinformatics, 28, 878–879.

Liu, Y. and B. Schmidt (2012): "Long read alignment based on maximal exact match seeds," Bioinformatics, 28.

Lucas Lledó, J. I. and M. Cáceres (2013): "On the power and the systematic biases of the detection of chromosomal inversions by paired-end genome sequencing." PloS one, 8, e61292, URL http://dx.plos.org/10.1371/journal.pone.0061292http://www.ncbi.nlm.nih.gov/pubmed/23637806http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3634047.

Lunter, G. and M. Goodson (2011): "Stampy: A statistical algorithm for sensitive and fast mapping of Illumina sequence reads," Genome Research, 21, 936–939.

Lupski, J. R. (2015): "Structural variation mutagenesis of the human genome: impact on disease and evolution," Environ. Mol. Mutagen., 56, URL https://doi.org/10.1002/em.21943.

Macintyre, G., B. Ylstra, and J. D. Brenton (2016): "Sequencing Structural Variants in Cancer for Precision Therapeutics," Trends in Genetics, 32, 530–542, URL http://www.ncbi.nlm.nih.gov/pubmed/27478068http://linkinghub.elsevier.com/retrieve/pii/S0168952516300701.

Mahmoud, M., N. Gobet, D. I. Cruz-Dávalos, N. Mounier, C. Dessimoz, and F. J. Sedlazeck (2019): "Structural variant calling: The long and the short of it," .

Mardis, E. R. (2017): "DNA sequencing technologies: 2006?2016," Nature Protocols, 12, 213–218, URL http://www.ncbi.nlm.nih.gov/pubmed/28055035http://www.nature.com/doifinder/10.1038/nprot.2016.182.

Merker, J. D., A. M. Wenger, T. Sneddon, M. Grove, Z. Zappala, L. Fresard, D. Waggott, S. Utiramerur, Y. Hou, K. S. Smith, S. B. Montgomery, M. Wheeler, J. G. Buchan, C. C. Lambert, K. S. Eng, L. Hickey, J. Korlach, J. Ford, and E. A. Ashley (2017): "Long-read genome sequencing identifies causal structural variation in a Mendelian disease." Genetics in medicine : official journal of the American College of Medical Genetics, URL http://www.nature.com/doifinder/10.1038/gim.2017.86http://www.ncbi.nlm.nih.gov/pubmed/28640241.

Metzker, M. L. (2010): "Sequencing technologies the next generation," .

Mills, R. E., K. Walter, C. Stewart, R. E. Handsaker, K. Chen, C. Alkan, A. Abyzov, S. C. Yoon, K. Ye, R. K. Cheetham, A. Chinwalla, D. F. Conrad, Y. Fu, F. Grubert, I. Hajirasouliha, F. Hormozdiari, L. M. Iakoucheva, Z. Iqbal, S. Kang, J. M. Kidd, M. K. Konkel, J. Korn, E. Khurana, D. Kural, H. Y. K. Lam, J. Leng, R. Li, Y. Li, C.-Y. Lin, R. Luo, X. J. Mu, J. Nemesh, H. E. Peckham, T. Rausch, A. Scally, X. Shi, M. P. Stromberg, A. M. Stütz, A. E. Urban, J. A. Walker, J. Wu, Y. Zhang, Z. D. Zhang, M. A. Batzer, L. Ding, G. T. Marth, G. McVean, J. Sebat, M. Snyder, J. Wang, K. Ye, E. E. Eichler, M. B. Gerstein, M. E. Hurles, C. Lee, S. A. McCarroll, J. O. Korbel, and 1000 Genomes Project (2011): "Mapping copy number variation by population-scale genome sequencing," Nature, 470, 59–65, URL http://www.ncbi.nlm.nih.gov/pubmed/21293372http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3077050http://www.nature.com/doifinder/10.1038/nature09708.

Mühle, C., M. Zenker, N. Chuzhanova, and H. Schneider (2007): "Recurrent inversion with concomitant deletion and insertion events in the coagulation factor VIII gene suggests a new mechanism for X-chromosomal rearrangements causing hemophilia A." Human mutation, 28, 1045, URL http://doi.wiley.com/10.1002/humu.9506http://www.ncbi.nlm.nih.gov/pubmed/17823971.

Nattestad, M. (2017): "Complex rearrangements and oncogene amplifications revealed by long-read DNA and RNA sequencing of a breast cancer cell line,"

BioRxiv.

Nielsen, R., J. Paul, A. Albrechtsen, and Y. Song (2011): "Genotype and SNP calling from next-generation sequencing data," Nature Reviews Genetics, 12, 443–451.

Olson, N. D., S. P. Lund, R. E. Colman, J. T. Foster, J. W. Sahl, J. M. Schupp, P. Keim, J. B. Morrow, M. L. Salit, and J. M. Zook (2015): "Best practices for evaluating single nucleotide variant calling methods for microbial genomics," .

Pavlopoulos, G. A., A. Oulas, E. Iacucci, A. Sifrim, Y. Moreau, R. Schneider, J. Aerts, and I. Iliopoulos (2013): "Unraveling genomic variation from next generation sequencing data," BioData Mining, 6, 13, URL http://www.ncbi.nlm.nih.gov/pubmed/23885890http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3726446.

Pfeifer, S. P. (2017): "From next-generation resequencing reads to a high-quality variant data set," Heredity, 118, 111–124, URL http://www.ncbi.nlm.nih.gov/pubmed/27759079http://www.nature.com/doifinder/10.1038/hdy.2016.102.

Quinlan, A. and I. Hall (2010): "BEDTools: A flexible suite of utilities for comparing genomic features," Bioinformatics, 26, 841–842.

Rang, F. J., W. P. Kloosterman, and J. de Ridder (2018): "From squiggle to basepair: computational approaches for improving nanopore sequencing read accuracy," Genome Biology, 19, 90, URL https://genomebiology.biomedcentral.com/articles/10.1186/s13059-018-1462-9.

Rausch, T., T. Zichner, A. Schlattl, A. M. Stütz, V. Benes, and J. O. Korbel (2012): "DELLY: structural variant discovery by integrated paired-end and split-read analysis." Bioinformatics (Oxford, England), 28, i333–i339, URL https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/bts378http://www.ncbi.nlm.nih.gov/pubmed/

22962449http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3436805.

Reinert, K., B. Langmead, D. Weese, and D. J. Evers (2015): "Alignment of Next-Generation Sequencing Reads," Annual Review of Genomics and Human Genetics, 16, 133–151, URL http://www.ncbi.nlm.nih.gov/pubmed/25939052http://www.annualreviews.org/doi/10.1146/annurev-genom-090413-025358.

Rescheneder, P., A. Von Haeseler, and F. Sedlazeck (2012): "MASon: Million alignments in seconds: A platform independent pairwise sequence alignment library for next generation sequencing data," in BIOINFORMATICS 2012 - Proceedings of the International Conference on Bioinformatics Models, Methods and Algorithms, 195–201.

Reuter, J., D. V. Spacek, and M. Snyder (2015): "High-Throughput Sequencing Technologies," Molecular Cell, 58, 586–597.

Rhie, A., S. A. McCarthy, O. Fedrigo, J. Damas, G. Formenti, S. Koren, M. Uliano-Silva, W. Chow, A. Fungtammasan, J. Kim, C. Lee, B. J. Ko, M. Chaisson, G. L. Gedman, L. J. Cantin, F. Thibaud-Nissen, L. Haggerty, I. Bista, M. Smith, B. Haase, J. Mountcastle, S. Winkler, S. Paez, J. Howard, S. C. Vernes, T. M. Lama, F. Grutzner, W. C. Warren, C. N. Balakrishnan, D. Burt, J. M. George, M. T. Biegler, D. Iorns, A. Digby, D. Eason, B. Robertson, T. Edwards, M. Wilkinson, G. Turner, A. Meyer, A. F. Kautt, P. Franchini, H. W. Detrich, H. Svardal, M. Wagner, G. J. Naylor, M. Pippel, M. Malinsky, M. Mooney, M. Simbirsky, B. T. Hannigan, T. Pesout, M. Houck, A. Misuraca, S. B. Kingan, R. Hall, Z. Kronenberg, I. Sović, C. Dunn, Z. Ning, A. Hastie, J. Lee, S. Selvaraj, R. E. Green, N. H. Putnam, I. Gut, J. Ghurye, E. Garrison, Y. Sims, J. Collins, S. Pelan, J. Torrance, A. Tracey, J. Wood, R. E. Dagnew, D. Guan, S. E. London, D. F. Clayton, C. V. Mello, S. R. Friedrich, P. V. Lovell, E. Osipova, F. O. Al-Ajli, S. Secomandi, H. Kim, C. Theofanopoulou, M. Hiller, Y. Zhou, R. S. Harris, K. D. Makova, P. Medvedev, J. Hoffman, P. Masterson, K. Clark, F. Martin,

K. Howe, P. Flicek, B. P. Walenz, W. Kwak, H. Clawson, M. Diekhans, L. Nassar, B. Paten, R. H. Kraus, A. J. Crawford, M. T. P. Gilbert, G. Zhang, B. Venkatesh, R. W. Murphy, K. P. Koepfli, B. Shapiro, W. E. Johnson, F. Di Palma, T. Marques-Bonet, E. C. Teeling, T. Warnow, J. M. Graves, O. A. Ryder, D. Haussler, S. J. OBrien, J. Korlach, H. A. Lewin, K. Howe, E. W. Myers, R. Durbin, A. M. Phillippy, and E. D. Jarvis (2021): "Towards complete and error-free genome assemblies of all vertebrate species," Nature, 592.

Robert, F. and J. Pelletier (2018): "Exploring the Impact of Single-Nucleotide Polymorphisms on Translation," .

Rovelet-Lecrux, A., D. Hannequin, G. Raux, N. L. Meur, A. Laquerrière, A. Vital, C. Dumanchin, S. Feuillette, A. Brice, M. Vercelletto, F. Dubas, T. Frebourg, and D. Campion (2006): "APP locus duplication causes autosomal dominant early-onset Alzheimer disease with cerebral amyloid angiopathy," Nature Genetics, 38, 24–26, URL http://www.ncbi.nlm.nih.gov/pubmed/16369530http://www.nature.com/doifinder/10.1038/ng1718.

Schaeffer, S. W. (2016): "Genome Organization, Evolution of," in Encyclopedia of Evolutionary Biology.

Sebat, J., B. Lakshmi, J. Troge, J. Alexander, J. Young, P. Lundin, S. Månér, H. Massa, M. Walker, M. Chi, N. Navin, R. Lucito, J. Healy, J. Hicks, K. Ye, A. Reiner, T. C. Gilliam, B. Trask, N. Patterson, A. Zetterberg, and M. Wigler (2004): "Large-Scale Copy Number Polymorphism in the Human Genome," Science, 305, 525–528, URL http://www.ncbi.nlm.nih.gov/pubmed/15273396http://www.sciencemag.org/cgi/doi/10.1126/science.1098918.

Sedlazeck, F. J., P. Rescheneder, and A. von Haeseler (2013): "NextGenMap: fast and accurate read mapping in highly polymorphic genomes," Bioinformatics, 29, 2790–2791, URL https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btt468.

Shimojima, K., T. Mano, M. Kashiwagi, T. Tanabe, M. Sugawara, N. Okamoto, H. Arai, and T. Yamamoto (2012): "Pelizaeus-Merzbacher disease caused by a duplication-inverted triplication-duplication in chromosomal segments including the PLP1 region." European journal of medical genetics, 55, 400–3, URL http://linkinghub.elsevier.com/retrieve/pii/S1769721212000833http://www.ncbi.nlm.nih.gov/pubmed/22490426.

Sohn, J. I. and J. W. Nam (2018): "The present and future of de novo whole-genome assembly," Briefings in Bioinformatics, 19.

Sović, I., M. Šikić, A. Wilm, S. N. Fenlon, S. Chen, and N. Nagarajan (2016): "Fast and sensitive mapping of nanopore sequencing reads with GraphMap." Nature communications, 7, 11307, URL http://www.nature.com/doifinder/10.1038/ncomms11307http://www.ncbi.nlm.nih.gov/pubmed/27079541http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4835549.

Stefanie Tauber (2013): Exploring the transcriptome: innovative methods for analyzing RNA-Seq data, Ph.D. thesis, Universität Wien.

Stephens, Z. D., S. Y. Lee, F. Faghri, R. H. Campbell, C. Zhai, M. J. Efron, R. Iyer, M. C. Schatz, S. Sinha, and G. E. Robinson (2015): "Big data: Astronomical or genomical?" PLoS Biology, 13.

Sudmant, P. H. (2015): "An integrated map of structural variation in 2,504 human genomes," Nature, 526, URL https://doi.org/10.1038/nature15394.

Tarasov, A., A. Vilella, E. Cuppen, I. Nijman, and P. Prins (2015): "Sambamba: Fast processing of NGS alignment formats," Bioinformatics, 31, 2032–2034.

Tattini, L., R. DAurizio, and A. Magi (2015): "Detection of Genomic Structural Variants from Next-Generation Sequencing Data," Frontiers in Bioengineering and Biotechnology, 3, 92, URL http://www.ncbi.nlm.nih.gov/pubmed/26161383http://www.pubmedcentral.nih.gov/articlerender.fcgi?

artid=PMC4479793http://journal.frontiersin.org/Article/10.3389/fbioe.2015.00092/abstract.

Teo, S. M., Y. Pawitan, C. S. Ku, K. S. Chia, and A. Salim (2012): "Statistical challenges associated with detecting copy number variations with next-generation sequencing." Bioinformatics (Oxford, England), 28, 2711–8, URL https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/bts535http://www.ncbi.nlm.nih.gov/pubmed/22942022.

Weischenfeldt, J., O. Symmons, F. Spitz, and J. O. Korbel (2013): "Phenotypic impact of genomic structural variation: insights from and for human disease." Nature reviews. Genetics, 14, 125–38, URL http://dx.doi.org/10.1038/nrg3373.

Wick, R. R. and K. E. Holt (2019): "Benchmarking of long-read assemblers for prokaryote whole genome sequencing," F1000Research, 8.

Xiao, C. L. (2017): "MECAT: fast mapping, error correction, and de novo assembly for single-molecule sequencing reads," Nat. Methods, 14, URL https://doi.org/10.1038/nmeth.4432.

Yang, X., W. P. Lee, K. Ye, and C. Lee (2019): "One reference genome is not enough," Genome Biology, 20.

Zichner, T., D. A. Garfield, T. Rausch, A. M. Stütz, E. Cannavó, M. Braun, E. E. M. Furlong, and J. O. Korbel (2013): "Impact of genomic structural variation in Drosophila melanogaster based on population-scale sequencing." Genome research, 23, 568–79, URL http://genome.cshlp.org/cgi/doi/10.1101/gr.142646.112http://www.ncbi.nlm.nih.gov/pubmed/23222910http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3589545.

Zimin, A. V., D. R. Smith, G. Sutton, and J. A. Yorke (2008): "Assembly

reconciliation," Bioinformatics, 24, 42–45, URL http://www.ncbi.nlm.nih.gov/pubmed/18057021https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btm542.

Zook, J. M., B. Chapman, J. Wang, D. Mittelman, O. Hofmann, W. Hide, and M. Salit (2014): "Integrating human sequence data sets provides a resource of benchmark SNP and indel genotype calls." Nature biotechnology, 32, 246–51, URL http://www.nature.com/doifinder/10.1038/nbt.2835http://www.ncbi.nlm.nih.gov/pubmed/24531798.

Appendix A

Supplementary Material to Chapter 3

A.1 Supplementary tables

Data	Genome	Technology	Reads		Source
			number [mio]	length [bp]	
R_1	human	Illumina	29.6	35	SRA
R_2	human	Illumina	10.4	76	SRA
R_3	human	Illumina	10.0	101	SRA
R_4	human	Ion Torrent	6.1	5-396	Life Tech.
R_5	human	454	2.8	51-4,935	SRA
S_1	human	Illumina	20.0	150	Mason
S_2	human	Illumina	20.0	250	Mason
S_3	*A thaliana*	Illumina	20.0	100	Mason
S_4	*D melanogaster*	Illumina	20.0	100	Mason
A_0	*A thaliana*	Illumina	5.0	100	Mason
A_1	*A thaliana*	Illumina	5.0	100	Mason
A_2	*A thaliana*	Illumina	5.0	100	Mason
A_3	*A thaliana*	Illumina	5.0	100	Mason
A_4	*A thaliana*	Illumina	5.0	100	Mason
A_5	*A thaliana*	Illumina	5.0	100	Mason
A_6	*A thaliana*	Illumina	5.0	100	Mason
A_7	*A thaliana*	Illumina	5.0	100	Mason
A_8	*A thaliana*	Illumina	5.0	100	Mason
A_9	*A thaliana*	Illumina	5.0	100	Mason
A_{10}	*A thaliana*	Illumina	5.0	100	Mason

Table A.1: Summary of the data sets used for benchmarking. All data sets are available on http://www.cibiv.at/software/ngm/data/

	BWA	**BWA-SW**	**Bowtie2**	**Stampy**	***NextGenMap*** *(CPU only)*	***NextGenMap*** *(+GPU)*
A_0	96.4(**2.2**)	97.6(2.3)	**97.7**(2.3)	**97.7**(2.3)	**97.7**(2.3)	**97.7**(2.3)
A_1	82.0(**1.9**)	96.9 (2.4)	97.5 (2.3)	**97.7** (2.3)	97.6 (2.4)	97.6 (2.4)
A_2	65.0(**1.6**)	95.0 (2.5)	96.2 (2.4)	**97.7** (2.3)	97.5 (2.5)	97.5 (2.5)
A_3	48.6(**1.2**)	91.4 (2.6)	92.8 (2.3)	**97.6** (2.4)	97.5 (2.5)	97.5 (2.5)
A_4	34.7(**0.8**)	86.2 (2.7)	86.5 (2.2)	**97.5** (2.5)	97.4 (2.6)	97.4 (2.6)
A_5	23.7(**0.6**)	79.6 (2.8)	77.5 (2.0)	**97.4** (2.5)	97.3 (2.7)	97.3 (2.7)
A_6	15.5(**0.4**)	71.9 (2.9)	66.5 (1.8)	**97.3** (2.6)	97.1 (2.8)	97.1 (2.8)
A_7	9.8(**0.3**)	63.7 (2.9)	54.8 (1.5)	**97.2** (2.7)	96.9 (3.0)	96.9 (3.0)
A_8	6.0(**0.2**)	55.4 (2.9)	43.2 (1.2)	**96.9** (2.8)	96.5 (3.2)	96.5 (3.2)
A_9	3.5(**0.1**)	47.3 (2.9)	32.8 (0.9)	**96.5** (3.0)	96.2 (3.4)	96.2 (3.4)
A_{10}	2.0(**0.1**)	39.8 (2.8)	24.1 (0.7)	**96.0** (3.1)	95.6 (3.7)	95.6 (3.7)

Table A.2: Percent of correctly (incorrectly) mapped reads for $A_0,\ldots,A_{10}$. For *NextGenMap* the results for CPU only and GPU version are shown (CPU/ GPU). We highlighted the highest percent of correctly and lowest percent of incorrectly mapped reads per data set.

	BWA	**BWA-SW**	**Bowtie2**	**Stampy**	***NextGenMap*** *(CPU only)*	***NextGenMap*** *(+GPU)*
A_0	3.7	10.2	3.6	36.9	2.3	**1.9**
A_1	8.7	10.2	3.8	54.1	2.7	**2.0**
A_2	9.8	9.9	3.9	66.8	2.6	**1.9**
A_3	10.1	9.4	3.8	74.2	2.5	**1.8**
A_4	8.6	9.2	3.6	80.4	2.6	**1.9**
A_5	8.1	8.7	3.4	83.8	2.8	**2.0**
A_6	5.9	8.1	3.1	89.1	3.0	**1.9**
A_7	4.9	7.9	3.0	88.7	3.9	**2.2**
A_8	4.0	7.2	2.7	86.3	4.5	**2.3**
A_9	3.2	6.9	**2.6**	87.0	9.5	3.2
A_{10}	**2.4**	6.4	**2.4**	88.1	11.7	3.9

Table A.3: Runtimes in minutes for $A_0,\ldots,A_{10}$. For *NextGenMap* the runtime for CPU only and GPU version are shown (CPU/ GPU). We highlighted the shortest runtime per data set.

A.2 Parameters for NextGenMap

In the following we give the full list of command line parameters of *NextGenMap* and their default values.

Parameter	Description
-c/–config <path>	Path to the config file. The config file contains all advanced options. If this parameter is omitted, default values will be used.
-r/–reference <path>	Path to the reference genome (format: FASTA, can be gzipped).
-q/–query <path>	Path to the read file. Valid input formats are: FASTA/Q (gzipped), SAM/BAM If the query file is omitted, NGM will only pre-process the reference.
-p/–paired	Input data is paired end. (default: off)
-I/–min-insert-size	The minimum insert size for paired end alignments (default: 0)
-X/–max-insert-size	The maximum insert size for paired end alignments (default: 1000)

Table A.4: List of input parameters for *NextGenMap*.

Parameter	Description
-o/–output <path>	Path to output file.
-b/–bam	Output BAM instead of SAM.
–hard-clip	Use hard instead of soft clipping for SAM output
–silent-clip	Hard clip reads but don't add clipping information to CIGAR string

Table A.5: List of output parameters for *NextGenMap*.

Parameter	Description
-t/–threads <int>	Number of candidate search threads
-s/–sensitivity <float>	A value between 0 and 1 that determines the number of possible mapping locations that will be evaluated with an sequence alignment. 0 means that all possible mapping locations will be evaluated 1 means that only the best possible mapping location(s) will be evaluated Higher values will reduce the runtime but also have a negative effect on mapping sensitivity. (default: estimated from input data)
-i/–min-identity <0-1>	All reads mapped with an identity lower than this threshold will be reported as unmapped (default: 0.65)
-R/–min-residues <int>	All reads mapped with lower than <int> residues will be reported as unmapped (default: 0.0)
-g/–gpu [int,...]	Use GPU(s) for alignment computation (GPU Ids are zero-based!). With -g or –gpu GPU 0 will be used. With -g 1 or –gpu 1 GPU 1 will be used. With -g 0,1 or –gpu 0,1 GPU 0 and 1 will be used. If -g/–gpu is omitted, alignments will be computed on the CPU
–bs-mapping	Enables bisulfite mapping (For bs-mapping kmer-skip will be applied to the reads instead of the reference sequence).
–bs-cutoff <int>	Max. number of Ts in a k-mer. All k-mers were the number of Ts is higher than <int> are ignored (default: 8)
-h/–help	Prints help and aborts program
-k/–kmer [10-14]	Kmer length in bases. (default: 13)
–kmer-skip <int>	Number of k-mers to skip when building the lookup table from the reference(default: 2)
–kmer-min <int>	Minimal number of k-mer hits required to consider a region a CMR. (default: 0)
–max-cmrs <int>	Reads that have more than <int> CMRs are ignored. (default: infinite)
-m/mode [0—1]	Alignment mode: 0 = local, 1 = semi-global. (default: 0)
-C/–max-consec-indels <int>	Maximum number of consecutive indels allowed.) (default: computed from input)

Table A.6: List of advanced parameters for *NextGenMap*.

Appendix B

Supplementary Material to Chapter 4

B.1 SV detection with Sniffles

Sniffles is a long read based structural variation (SV) caller capable of detecting deletions, tandem duplications, insertions, inversions, translocations as well as combinations (nested SV) of these five SV types. Sniffles incorporates multiple approaches for detecting small (by default: 30bp) to large (10kb+) SV. Furthermore, Sniffles implements multiple novel techniques to ensure a low false discovery rate. In the following, we give a detailed description how Sniffles works and the implemented novel mechanisms that lead to an enhancement in sensitivity and precision. Sniffles incorporates four major phases, which are all automatically executed during runtime (Supplementary Figure B.1). First, Sniffles quantifies characteristics of the input data such as the overall sequencing error rate, SNP rate, and the average alignment scores of the reads. Second, Sniffles uses within-alignment and split read information to call SVs. Finally, in steps 3 and 4 Sniffles infers genotype and phasing information from the SVs called in the previous steps. In the following we give a detailed description of the individual parts.

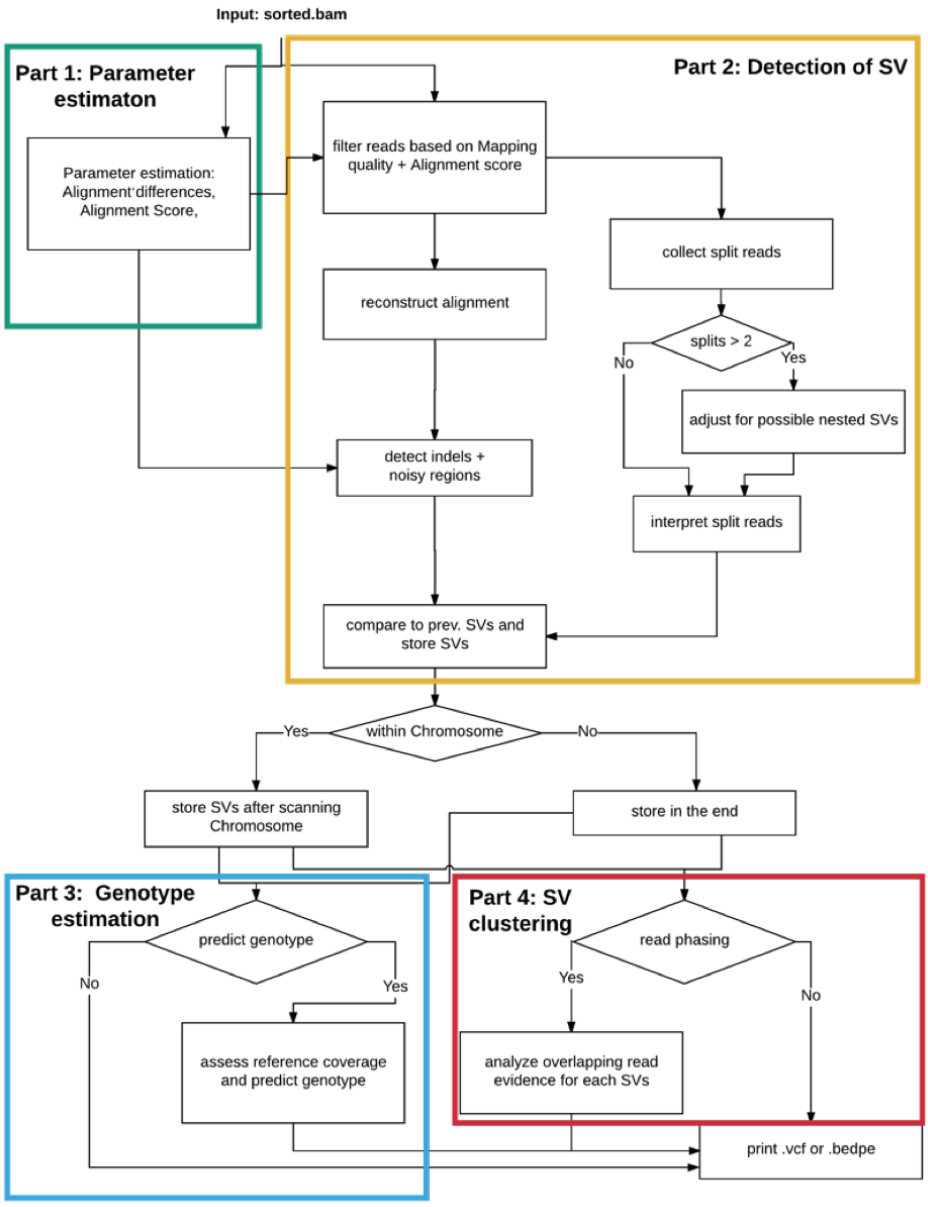


Figure B.1: Overview of Sniffles workflow. Sniffles incorporates four major steps. (1) First, it estimates required parameters from the mapped read file. (2) Second, it identifies SVs based on split read and alignment events. (3) Third, optionally it estimates the genotype of the called SVs. (4) Fourth, optionally it attempts to cluster SVs together based on the same set of reads that are overlapping (Part 4).

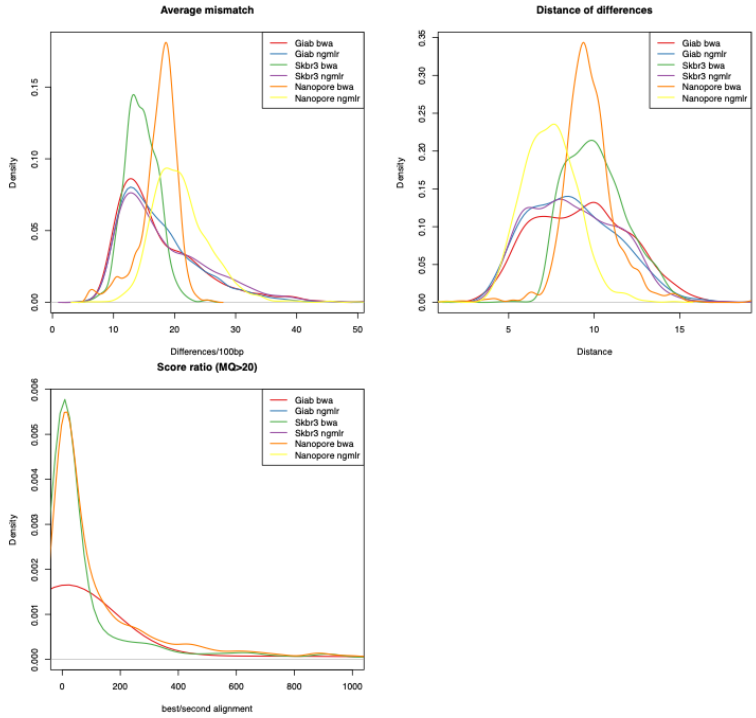


Figure B.2: Density Plots for the parameter estimation of two different PacBio data sets (Giab, SKBR3) and one Nanopore (NA12878).

B.1.1 Preprocessing and parameter estimation

The Sniffles SV calling algorithm depends on three key parameters:

1. The sequencing error observed in the reads, the average distance between indels or mismatches in the read alignments, and the 95th percentile of the number of mismatches and indels in a 100bp window. Sniffles uses these parameters to detect read alignments or parts of read alignments that show an increased mismatch/indel rate and therefore might overlap with a SV.
2. The ratio between the best and second best alignment scores for each read mapping. Sniffles uses this parameter to assess the reliability of a particular read mapping.

These parameters differ between genomes and sequencing technologies (see Supplementary Figure B.2). Therefore, Sniffles tries to estimate them from the input

data. To this end, Sniffles scans $10,000$ randomly chosen reads to obtain the distribution of these parameters. We emphasize that this is done solely to estimate these parameters and not designed to obtain a comprehensive view across the entire genome. The assumption is SVs are generally rare so that most of these reads should not overlap a SV and thus representing a solid baseline of these parameters.

B.1.2 Scanning for SVs

To reduce false positive SV calls Sniffles stringently filters for spurious read mappings. A read is discarded if it has a mapping quality lower than 20 (by default) or the ratio of its best and second best alignment score is less than 2, or its alignment score ratio is smaller than the minimum alignment score ratio computed by the parameter estimation step (see section B.1.1). Furthermore, a read is discarded if it shows more than 7 (by default) split read alignments or if none of the aligned parts of the read exceeds a length of 1 kbp (by default).
For each of the remaining reads, Sniffles performs the following four steps to detect SVs:

1. Sniffles scans the read alignments to detect smaller (¡1kb) insertions, deletions and regions with an increased number of mismatches and very short (1-5bp) indels. These noisy regions often indicate incorrect read mappings caused by SV.
2. Sniffles processes split read information to identify SV that cannot be represented in a single alignment (large indels, inversions, duplications, translocations). All SVs found during step 1 and 2 are stored in a self-balancing binary tree.
3. Sniffles traverses the binary tree to merge SV calls that were caused by the same SV.
4. After all reads were scanned for SVs, Sniffles identifies spurious SV calls and discards them.

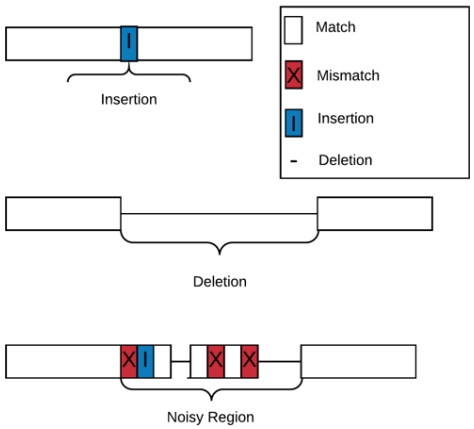


Figure B.3: Schematic of alignment events. Sniffles distinguishes between 3 different types: insertions, deletions and noisy regions. Noisy regions often indicate that a read overlaps a deletion or inversion.

Alignment analysis

To identify SVs within read alignments Sniffles first extracts the genomic position and the length of all mismatches and indels from the MD and CIGAR string of every read. All insertions and deletions longer then the user defined minimum SVs length (default: 30bp) are recorded as potential starting points of a SV. Next, we use a PlaneSweep algorithm to identify segments of the read that show an increased number of mismatches and small indels which we will refer to as noisy regions (see Supplementary Figure B.3 for an example).

PlaneSweep algorithms are widely used in genomics for determining for example the read coverage for each position of a reference genome. Instead of read start and end positions, here we use the genomic position of the mismatches and indels as start and define the end location of the interval as its length plus 100bp (by default) to allow for some noise in the position of the SV event. Our PlaneSweep algorithm is modified so that it takes these intervals as input and outputs candidate SV regions if the number of mismatches and indels at a certain position exceeds the maximum number of differences per 100bp computed in the parameter estimation step (section B.1.1). Thus, each of the coordinates identified by the PlaneSweep algorithm represent start locations of a segment along the read that shows an in-

creased accumulation of mismatches and short indels and therefore might overlap a SV. Subsequently, for each of previously stored regions, Sniffles attempts to enlarge the size in both directions. This is done until the distance between two differences on the read minus the length of the event is larger than the maximum difference distance allowed, as determined in the parameter estimation (section B.1.1).

Next, Sniffles tries to determine the SV type that most likely caused the noisy regions. To this end, Sniffles uses the number of mismatches (M), the maximum length of a single insertion (sl) and the maximum length of a single deletion event (dl) within the determined region. Sniffles reports an insertion if sl $>$ minimum SVs size (default: 30bp) and the sum of all insertion lengths observed in that region is at least twice as much as sum of the length of all deletions. Deletions are identified using a reciprocal rule. We required this more complicated rule set to distinguish clear indels from noisy regions in a read. Sniffles reports the region as being noisy if it is neither a distinct insertion nor a distinct deletion, but the sum of mismatches in this area is larger than the minimum allowed SV size (default 30bp). Such noisy regions are most commonly caused by deletions or inversions, especially from aligners other than NGMLR.

In addition to these three cases (insertions, deletions and noisy regions), Sniffles also checks whether the read was clipped by more than 2kbp (by default). This often indicates that the read partly spans an insertion. Furthermore, Sniffles utilizes a flag reported by NGMLR, which indicates if a read was clipped due to Ns in the reference genome or if it could not identify a region where the clipped sequence maps to. In this case we cannot determine the exact size and thus we set every insertion of this type to have a size of *NA*.

At the end of this scan, Sniffles knows the number of potential insertions, deletions and noisy regions in every read. If a read contains more than 3 noisy regions it is discarded and all SV calls from this read are removed. Otherwise, all remaining SVs are inserted into our binary tree to collect all potential SVs from all valid reads (see section B.1.2).

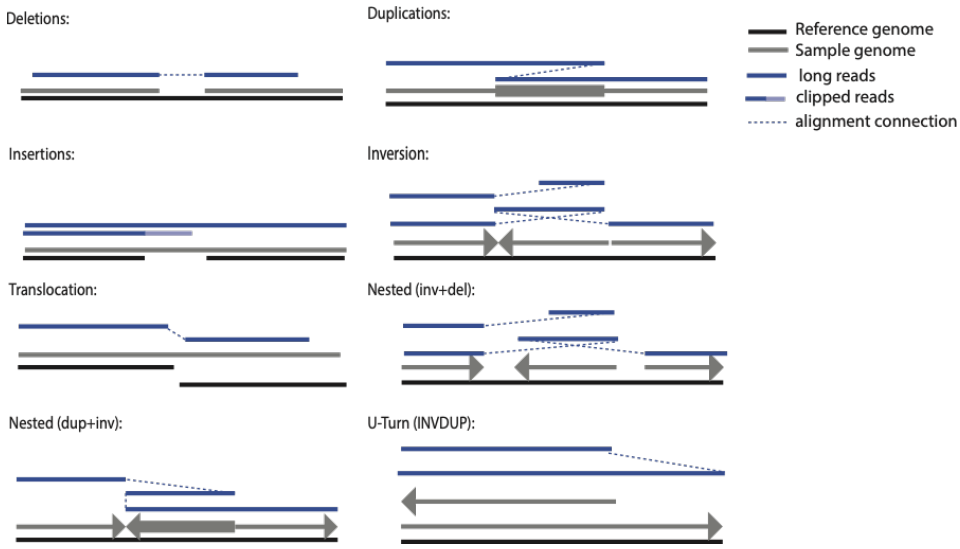


Figure B.4: Schematic illustration of the different SV types Sniffles can detect and the split reads can be used to detect these events.

Split read analysis

Reads spanning inversions and translocations can only be mapped by splitting the read and reporting two or more separate alignments (see Supplementary Figure B.4 for examples of split read alignments). These split read mappings are typically reported as separate SAM records (see SAM specifications by (Li et al., 2009)) in a SAM/BAM file. For each read one of the alignments (usually the longest) is marked as primary alignment. All the others are called supplementary alignments. To simplify parsing, BWA-MEM and NGMLR compute the SA tag for each SAM record. The SA tag of the primary alignment record contains all information about all supplementary alignments required for SV calling. Therefore, Sniffles only reads the primary alignment for each read from the BAM file and extracts supplementary alignment information from the SA tag.

Note that the SA tag does not provide a second-best alignment score, therefore Sniffles filter supplementary alignments only by MQ but not by alignment score ratio. Reads with more than N alignments (default $N = 7$, but user definable) are typically low quality reads and are therefore ignored. Furthermore, Sniffles ignores reads where non of the sub alignments are larger then 1kbp (by default) as these

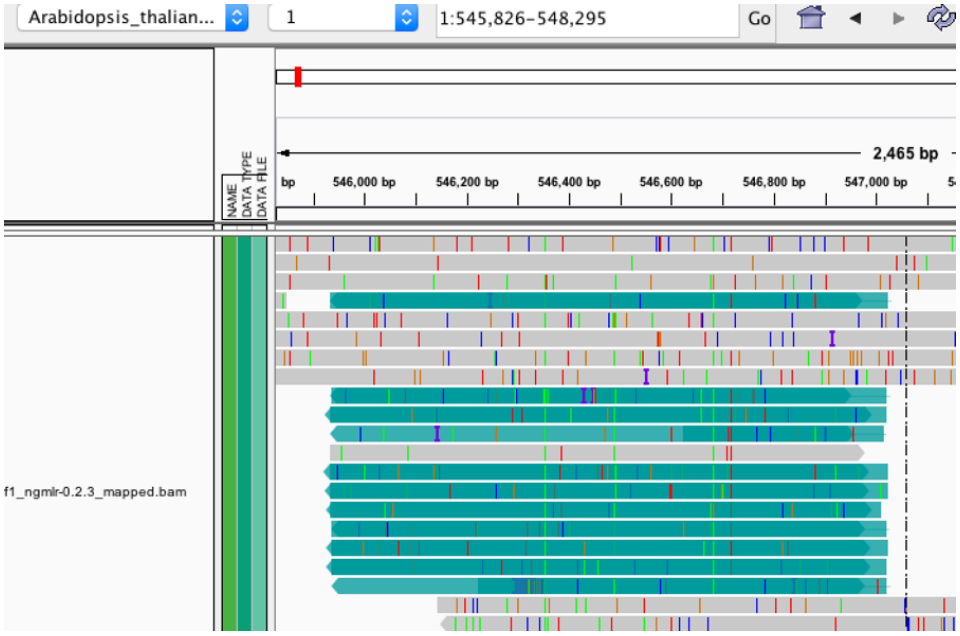


Figure B.5: Example INVDUP signal due to an error in base calling and/or subread segmentation. Note that one polymerase read (marked) is folded over 11 times on top of itself with in a 1 kbp region.

often sequencing artifacts (Supplementary Figure B.5). For all other reads, Sniffles extracts starts and end positions on the read and on the reference of the primary and supplementary alignments and stores them as intervals in a list ordered by the starting position on the read.

Next, Sniffles divides the reads into two categories: reads with two alignments that most probably span simple SVs, and reads with more alignments that potentially span nested SVs. For reads consisting of two alignments Sniffles applies the following rules to detect SVs:

1. If the two segments of the read are aligned to the same chromosome and share the same orientation then Sniffles compares their distances on the read level to their distances on the genome level. Sniffles calls an insertion if the distance on the read level minus the distance on the genome is larger than the minimum SV length (default 30bp). Sniffles calls a deletion if the distance on the genome level minus the distance on the read level is larger than the minimum SV length. If the distance on the reference is larger than

the minimum SV length and the distance on the read level is smaller than the minimum SV length, Sniffles calls a duplication.

2. If the two segments are on the same chromosome, but have different strands Sniffles calls an inversion. In case these two segments overlap by at least 200bp and at least 40% of the shorter segment length they are called an inverted duplication (”U-turn”). For inverted duplications, Sniffles requires one of the segments to be larger than 2kbp (by default). This is necessary to ignore certain base calling artifacts where a read is ”folded” multiple times onto itself (Supplemental Figure B.5)
3. If the two segments are aligned to different chromosomes Sniffles calls a translocation.

Reads that consist of more than two segments with a length larger than 200 bp and map to the same chromosome potentially cover a nested SV. For such reads, Sniffles applies separate rules: If the segments are overlapping and only one segment has a different strand then the other two, an inverted duplication is called. Furthermore, to account for inversion flanked with indels we determine if one segment has a different strand then the other two flanking it and the overlap or the distance of the segments in read space vs. the reference space is more then the minimum SVs length. If this is the case, Sniffles introduces pseudo segments, which are 1bp long elements that ease the detection of such complicated inversion segments. This is necessary to generalize and distinguish this nested inversion,
Finally, all detected SVs are inserted into our binary tree of all detected SVs in all valid reads (see Section B.1.2).

Storing/Clustering of SVs

Sniffles use a self-balancing binary tree to store and merge SV calls. Each node in the tree represents a single SV. The SVs are sorted based on the start coordinate of each SV.
Each time Sniffles detects a read that supports a SV, Sniffles traverses the binary tree to see if that particular SV has been observed before. The current SV call is merged with an already known one if their types (e.g. deletion) are the same and

their breakpoints are within the maximum distance D. Sniffles automatically infers sensible values for D based on SV length and type (see section B.1.2). In addition, an upper bound for D can be specified by the user (by default: 1kbp). This tolerated distance (D) is necessary to account for imprecise alignments due to sequencing errors or sequence composition (e.g. microsatellites) or non-optimal score function of the mapper itself (Supplementary Figure B.8).

In the tree, each SV is represented by the coordinates that it was first found at. However, the coordinates from other reads supporting the same SV are stored as well. To store the SV type Sniffles uses a set of bit flags to enable a fast comparison between different SVs. Furthermore, the bit flags allow Sniffles to assign multiple types and additional information to a single SV, especially for nested SVs. For complex types, we allow inversions or deletions to be merged with a candidate SV as long as they agree on the coordinates. Furthermore, we allow insertions and tandem duplications to be merged since a tandem duplication is an insertion of the same element next to itself.

To account for multiple overlapping SVs or SVs in close proximity, especially if the genome is polyploid in this region as commonly observed in human cancers or plant genomes, Sniffles implements a more thorough tree search to assess whether the current SV has already been observed. Here, Sniffles starts at the current parental node and walks using an in-order traversal search through the sub tree to identify an already stored SV that would match the current one. Note that this does not significantly increase the runtime, since this procedure will generally only be performed on a very small subtree.

If Sniffles does not find the current SV in the tree, it adds it as a new leaf node. Each SV is stored together with the name of the read it was observed in, the strands, the start and stop position of the genome, the start and stop position on the read, the bit-flag for the type and information about the source (split reads, alignment event, noisy region).

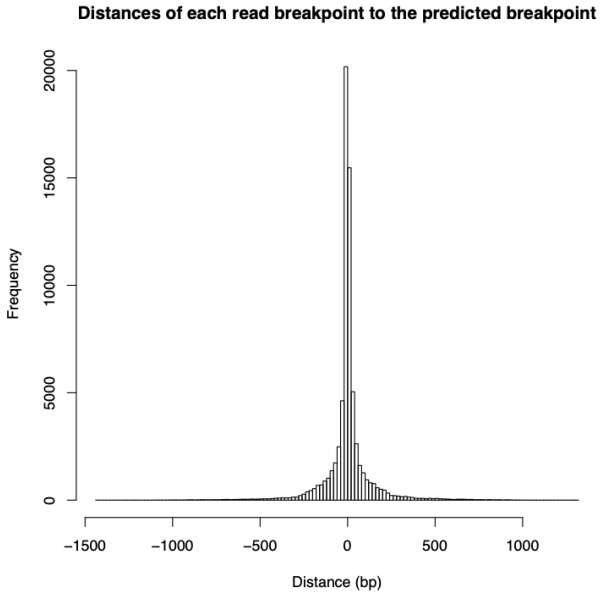


Figure B.6: Breakpoint distribution of clustered read events compared to the estimated breakpoint taken from real data in the SKBR3 data set.

Estimation of maximum distance between SVs

In the previous section, we described how Sniffles merges SVs based on their type and the distances between their breakpoints. This is necessary due to imprecise alignment breakpoints that can occur due to sequencing errors, complexity or sequence composition (e.g. microsatellites) or non-optimal score function of the mapper itself. Next Sniffles estimates the maximum distance D that should be allowed between genuine variants. If the distance is too large, it will merge distinct SV calls or spurious SVs stemming from sequencing artifacts in regions with high read coverage and falsely call a SV. Supplementary Figure B.8 shows examples of insertions caused by PacBio sequencing artefacts. Conversely, if the distance is too short, a genuine single SV could be called twice with slightly different breakpoints. In addition, read support for these SVs will be lower, potentially causing them to be filtered by the minimum read support threshold.

Sniffles attempts to balance these two scenarios, while letting the user decide about the maximum distance (d). We define the estimated length of an SV as $length\_est =$

$max(length(SV), min\_size * 2)$, where *min_size* is user defined (default: 30bp). This controls for very small events that are harder to place due to e.g. sequencing errors. The allowed distance between two SV are then computed as $d = min(length\_est(SV), d)$. This allows for an adaptive distance threshold while setting the upper bound according to the user. Translocations have an undefined size since they connect two chromosomes together. Thus, $d = d$ in this case. When comparing two SVs, Sniffles computes the allowed maximum distance (d) for both SVs and takes the smaller estimate for the comparison to avoid merging two distinct SV with different sizes.

Filtering and summarizing SVs

When Sniffles reaches the first read of a new chromosome it triggers the filtering and processing step for all the SVs observed so far. Only translocations are omitted as they might span a chromosome not yet processed by Sniffles. All other SV types identified so far are processed and printed. All translocations are summarized and filtered after the last read of the data set was processed.
For each SV, Sniffles counts the number of supporting reads and compares this to the user defined threshold (default: 10 reads). If the number of supporting reads is at least the threshold, Sniffles computes the most likely start and stop position. This is done by counting the number of reads supporting the same start position. If at least 5 reads share the same breakpoint position, then that value is used. Otherwise, Sniffles reports the average position of the breakpoints. This is done independently for start and stop breakpoints. For insertions Sniffles further computes in the same way the supported length of the insertion.

B.1.3 Detection of spurious SV calls

Usually, all the read alignments that support a SV call show very similar breakpoint positions. However, high sequencing error, repeat rich sequences, or low complexity regions of the genome can cause the breakpoints from different reads supporting the same SV to be scattered. One example are short randomly occurring insertions caused by sequencing or base calling errors. Especially in regions with

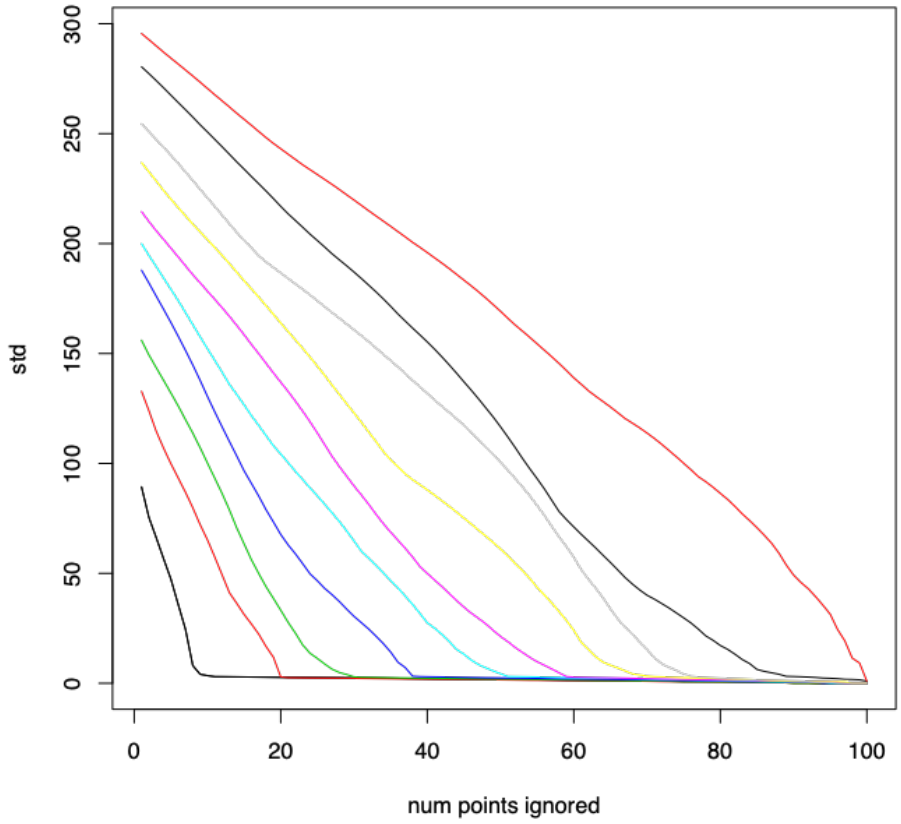


Figure B.7: Simulated breakpoints of a mix of a uniform distribution and a normal distribution. From left to right the concentration of uniform distributed breakpoints increases by 10%. The y axis reports the standard deviation (σ) given that we ignore the most outliers (x axis). In a typical truly identified SVs we would expect some alignments to break earlier or later depending on the sequencing error or sequence complexity at the region. However, a complete artificial signal should have a broader distribution even with a more stringed filtering.

high read coverage, the number of these random insertions might exceed the minimum read support causing Sniffles to report them as genuine SV. To filter out these phantom insertions Sniffles scans for abnormally noisy breakpoints. For a spurious SV call we expect the positions of the read breakpoints to be uniformly randomly distributed. In contrast, for genuine SVs we expect read breakpoints to be normally distributed (with a small standard deviation) around the correct breakpoints of the SV. Thus, we need to distinguish between a uniform/random distribution of read breakpoints and normal distributed read breakpoints even when the number of spanning reads is low. To this end, Sniffles computes the standard deviation (σ) of all read breakpoints that support a SV (separately for the start end the end breakpoints) as follows:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

where:

$$\mu = \frac{1}{N}(x_1 + ... + X_N)$$

or the most ($n > 5$) observed break point.

The idea is that σ will be small for normally distributed read breakpoints stemming from genuine SVs and large for random read breakpoints. A crucial factor for this analysis is read coverage. To determine how many read breakpoints are required to confidently differentiate between real and random breakpoints, we sampled genomic locations once from a uniform distribution and once from a normal distribution ($\sigma = 5$) and investigated different sample rations between those two.

Supplemental Figure B.9 shows the impact of coverage on the reliability of our standard deviation based filter. Given only 5 or more observed read breakpoints we can reliably distinguish between the uniform and the normal distribution. Thus, Sniffles uses the standard deviation of read breakpoints to filter out alignment artifacts or phantom SVs. However, even for genuine SV we sometimes observe outliers in the read break point distribution. These outliers artificially increase σ making it more difficult to distinguish them from random breakpoints. Supplemental Figure B.7 shows an example, of different mixtures of noisy breakpoints and precise break-

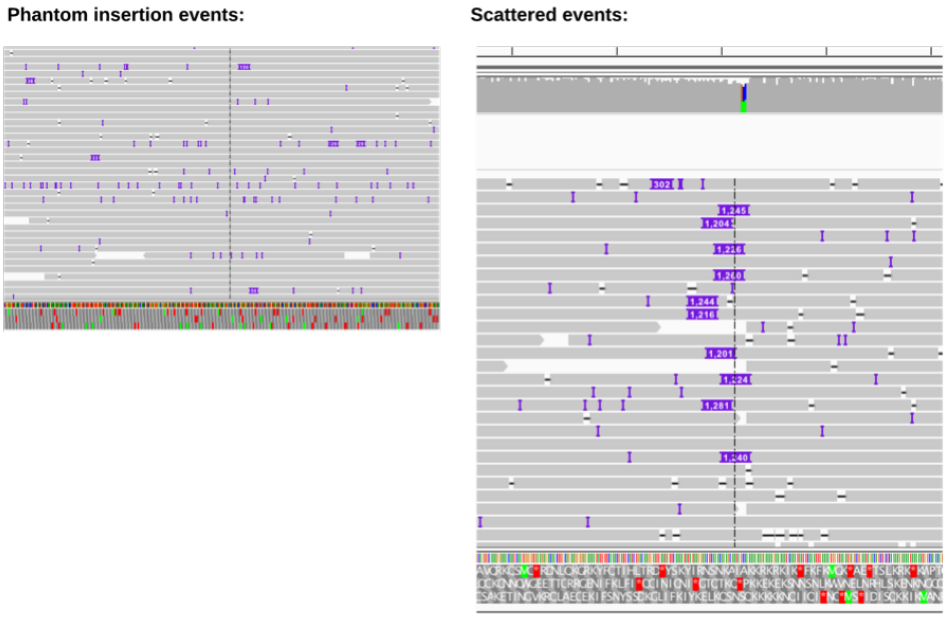


Figure B.8: Comparison of phantom insertions (region of $\sim$ 800 bp) and scattered insertions (region of $\sim$ 200 bp) using IGV. We define phantom insertions as random events that happen due to the base calling of PacBio. In high coverage regions these events sometimes cluster to form a signal. Sniffles detects and erases such calls. We define a scattered insertion as a real SV event, that has imprecise breakpoints. This is often due to the sequencing error occurring near the breakpoints. Sniffles cluster these events together and allows for a wobble on the breakpoints according to the size of the SV.

points. Here we observe a large impact of already 10-20% of the reads being more disturbed on σ. This has a direct impact on the comparison between the phantom events and real SV as shown in Supplemental Figure B.8.
To account for this, we perform 1st and 4th quintile filtering. Supplementary Figure B.10 shows the difference between σ with and without quintile filtering for genuine SVs from the SKBR3 data set. If the set of breakpoints is originally approximately a random/uniform distribution, σ will not change when doing quintile filtering (see Supplementary Figure B.4). However, for real SVs with a small fraction of outlier breakpoints σ will dramatically reduce (even to 0) when applying out filter (see Supplementary Figure B.7).
Next, we need to compare the trimmed-σ to an expected σ given the type and length of the SV. Here, we use a uniform distribution, with σ equal to the length of the region $(d) * sqrt(1/12)$. Since we compare it to the σ of a trimmed distribution, we need to correct this threshold accordingly. Through simulations of random variables within d iterating 1000 times we identified a ratio of filtered vs. not filtered of 1.99949. Thus, a SV is filtered out when the filtered $sd > d * (sqrt(1/12)/2)$ as this is most likely a set of spurious events rather than a true SV.

Detection of falsely merged SV calls

In section B.1.2 we described how Sniffles merges SVs of the same type and with breakpoints in close proximity to account for sequencing error, repeat rich sequences, and low complexity regions. However, sometimes this causes two separate SVs in close proximity to be falsely merged. To detect such falsely merges SV calls, Sniffles evaluates if the read breakpoint positions associated with a single SV follow a bimodal distribution. Supplementary Figure B.11 shows such a case for two merged translocation breakpoints. There are multiple ways to test for bimodal distributions such as evaluating the skewness and kurtosis, or Hartigan's Dip Test Statistic for Unimodality (Hartigan and Hartigan, 1985). Nevertheless, the statistics require a minimum number of observations (read breakpoints in our case) to be reliable, which is higher than the read coverage in a typical dataset. Therefore, they are not applicable here, and instead we apply a heuristic that we have found

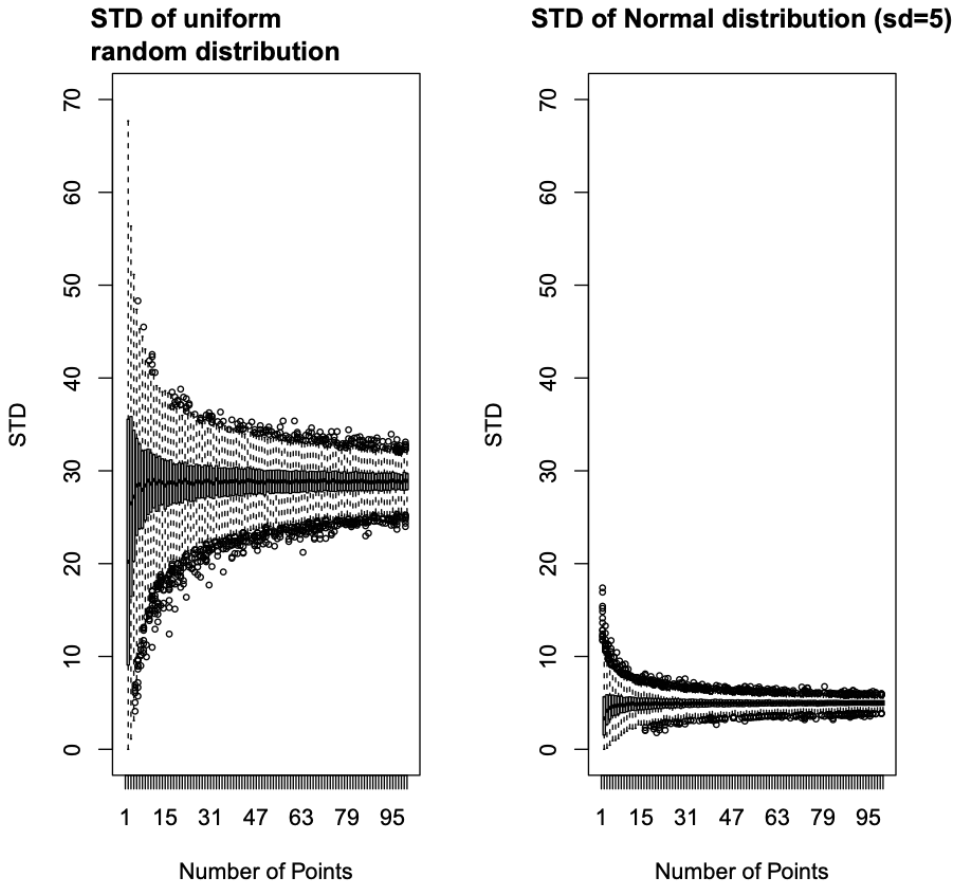


Figure B.9: Box-and-whisker plot of standard deviation given different number of points/coverage levels from 1-100. Left: Uniform random variables were simulated from an interval of 100bp. Right: Random variables were simulated from a normal distribution with $\mu = 0$ and $\sigma = 5$. Note that both plots converge to the expected $\sigma = 28.8$ for the uniformly random distribution and $\sigma = 5$ from the normal distribution. For each point we run the sampling 1,000 times. Box plots where generated using the default values from R.

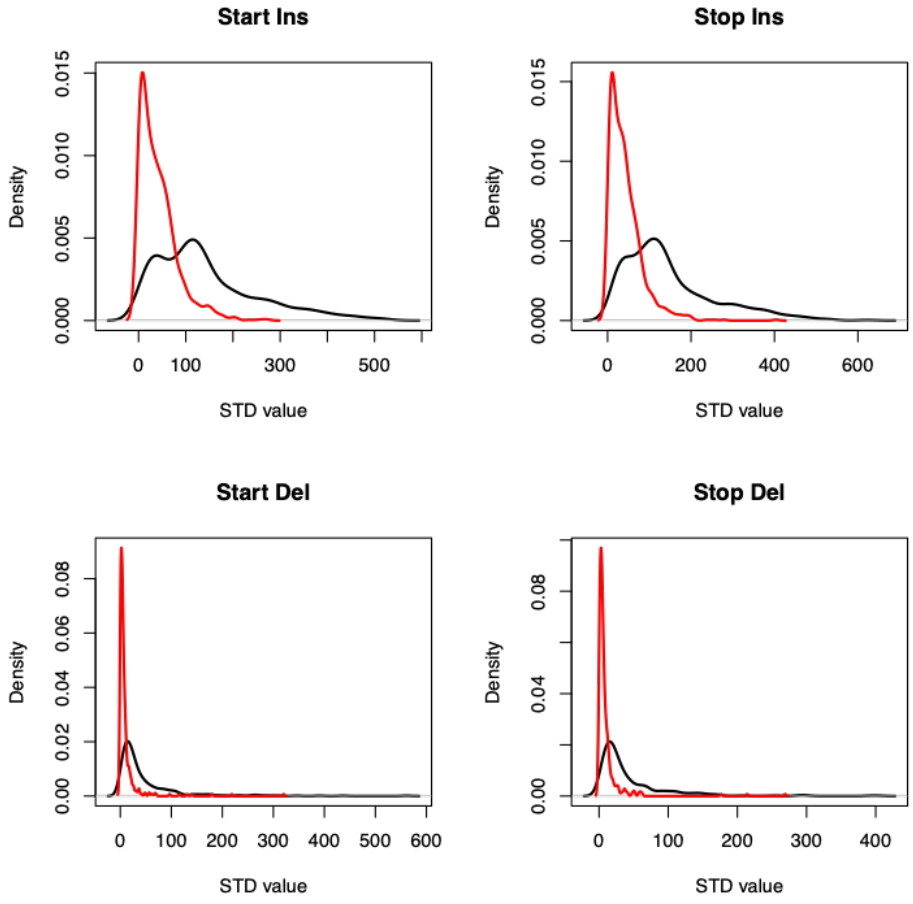


Figure B.10: Filtered (red) and unfiltered (black) breakpoint distribution over insertion and deletions on chr17 of the SKBR3 data set. As one can see the filtering enables a more precise prediction of noise vs. precise events.

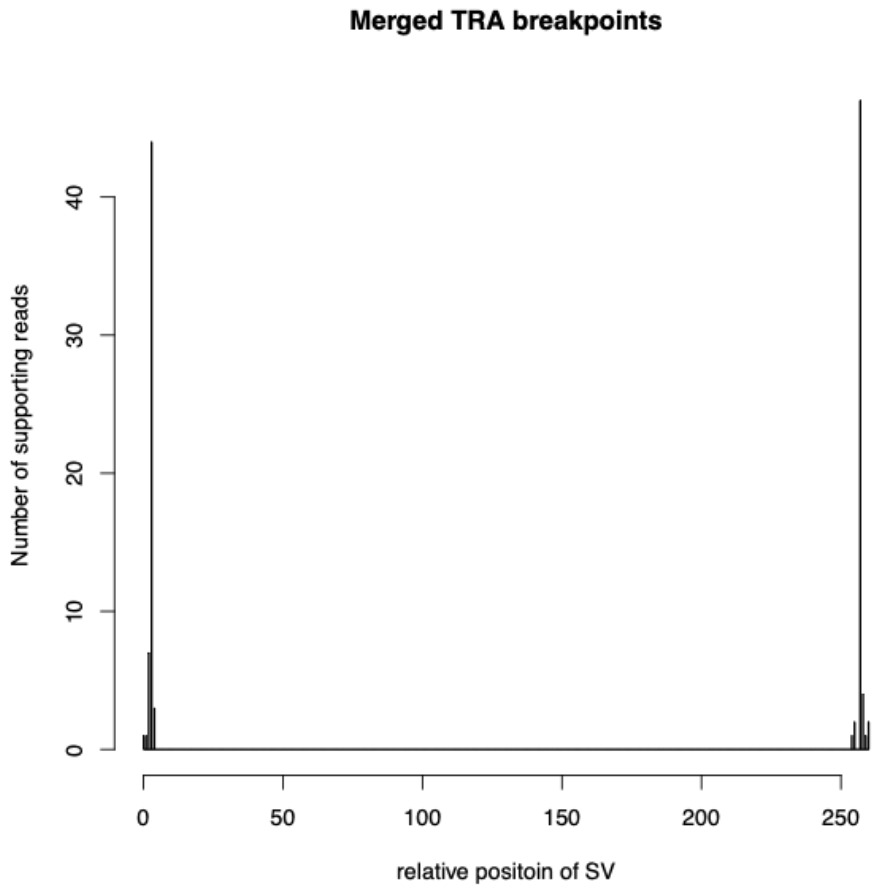


Figure B.11: Example of two translocation breakpoints merged to one event leading to a bimodal distribution in the breakpoint position.

performs well in practice.
Our heuristic approach first clusters the breakpoints within the minimum size of an SVs together. Next, Sniffles counts the number of clusters that have more breakpoints clustered than the minimal number of reads supporting a SV. If this is the case, we replace the original entry with two new entries according to the clustering.

B.1.4 Genotyping

To infer genotype and allele frequency for SV calls, Sniffles also needs to access information about reads that do not support SVs. Therefore, while scanning the alignments Sniffles also records the start and location of reads that do not support a SV, but otherwise match the filter requirements (score ratio and MQ quality) in a separate binary file. After all alignments are scanned, Sniffles analyzes the file of read placements and check for every read if it overlaps one or more SV calls and a read counter is increased for the respective SV call. This is efficiently computed using Sniffles self-balancing binary tree that is used to store all SV calls. Finally, Sniffles uses this information to estimate the genotype based on the allele frequency of the SVs and prints the result to the VCF file.

B.1.5 Phasing and read clustering

If the phasing/read clustering mode is activated, Sniffles records the number of reads that support each SV as well as their read IDs. This is computed using a hash table with the read IDs as key and the variant IDs as value. In addition, Sniffles scans the hash table and groups all variants spanned by a single read. This is computed by storing the lowest observed variant ID for each read in cases there are multiple variants assigned to one read name. For each so detected variant, Sniffles updates the hash table to set the variant ID to the minimum observed variant ID of the subgroup to indicate their clustering. In addition, Sniffles stores the number of reads that carry the same variant IDs. Given a user definable threshold (default: 1) on the minimum number of reads supporting the same two variants, Sniffles cluster the variant IDs together. The clustered IDs are indicated by the same number plus

a _ with a running number from 0 to the number of variants associated to aid in downstream analysis and further phasing.

B.2 Coverage based filtering of Arabidopsis CVI

We used Sambamba (v0.6.6) (Tarasov et al., 2015) to compute the base pair coverage across the CVI PacBio mapping. Next we used SURVIVOR (option 23) to identify and cluster regions with zero coverage within 10kb intervals. We excluded these regions from further analysis in the Col-0 x CVI F1, as the zero coverage indicates these regions are specific to Col-0.

B.3 Determining the effect of genome coverage on SV calling

After evaluating SV calling performance for Sniffles and NGMLR on high coverage datasets, we tested the effect of reducing genome coverage. To this end we first established an upper bound for how many SVs can be detected using a certain coverage using a naive simulation. This simulation is implemented in the SURVIVOR toolkit. In brief, we first simulate reads for a genome given a specified coverage and read length. Next, we simulated a set of SVs (700 for this experiment). For each breakpoint we count how many simulated reads overlap with this breakpoint by 50bp on each side and call these overlapping reads. Finally, we count the number of overlapping reads for each break point. Break points with > 5 overlapping reads are considered detected, break points with fewer we call undetected. The assumption is that a read that overlaps a break point by 50bp at each side can be used to successfully call the break point and that 5 overlapping reads give sufficient information to distinguish real break points from sequencing artifacts. We note that both assumption do not hold true in reality however it gives us an upper bound for SV calling recall.

Next, we compared this theoretical upper bound for recall to results obtained from genuine sequencing data. To this end we used seqtk[1] (version: 1.1-r93-dirty) to subsample the raw read data to 5x, 10x, 15x, 20x, 30x using the average read length of 4,334 and 9,872 for NA12878 and SKBR3 Pacbio datasets, respectively. For Nanopore we used the average read length of 6,432bp. Next, we mapped the subsampled read files using NGMLR and called SVs using Sniffles with different parameters for minimum read support (s: 1,2,3,4,5,6,7,8,9,10). Finally, we compute precision and recall for all call-sets by using the SV calls we got from the full datasets as our truth-set.

B.4 Supplementary tables

Program	Techn.	Type	Length	Precise	Indicated	Forced	Fragmented	Trimmed
BLASR	PacBio	DEL,INS	100	4.10	41.67	51.97	0.00	0.00
BLASR	PacBio	DEL,INS	250	7.12	65.72	24.48	0.00	0.00
BLASR	PacBio	DEL,INS	500	12.59	74.65	3.50	0.00	1.50
BLASR	PacBio	DEL,INS	1000	13.45	68.47	2.20	0.00	8.59
BLASR	PacBio	DEL,INS	5000	3.61	40.76	3.19	0.00	38.23
BLASR	PacBio	DEL,INS	10000	9.20	39.67	1.24	0.00	46.74
BLASR	PacBio	DEL,INS	50000	17.01	58.76	0.19	0.00	23.97
BLASR	PacBio	DUP	100	0.00	39.74	59.57	0.00	0.00
BLASR	PacBio	DUP	250	0.00	71.50	29.92	0.00	0.00
BLASR	PacBio	DUP	500	0.00	81.93	12.16	0.00	3.74
BLASR	PacBio	DUP	1000	0.00	71.66	6.89	0.00	18.69
BLASR	PacBio	DUP	5000	0.71	5.14	5.78	0.00	50.63
BLASR	PacBio	DUP	10000	8.00	18.58	1.96	0.00	15.73
BLASR	PacBio	DUP	50000	35.43	66.70	0.35	0.00	0.00
BLASR	PacBio	TRA	100	0.00	0.00	98.67	0.00	0.00
BLASR	PacBio	TRA	250	0.00	0.00	98.60	0.00	0.00
BLASR	PacBio	TRA	500	2.04	0.42	96.61	0.00	1.04
BLASR	PacBio	TRA	1000	15.68	0.87	81.28	0.00	1.70
BLASR	PacBio	TRA	5000	73.20	8.03	13.48	0.00	5.29
BLASR	PacBio	TRA	10000	93.10	3.26	0.46	0.00	3.01
BLASR	PacBio	TRA	50000	96.95	0.00	0.73	0.00	2.17

[1] `https://github.com/lh3/seqtk`

BLASR	PacBio	INV	100	0.00	0.00	99.19	0.00	0.00
BLASR	PacBio	INV	250	0.00	0.00	99.23	0.00	0.00
BLASR	PacBio	INV	500	0.80	0.88	96.97	0.00	0.80
BLASR	PacBio	INV	1000	7.95	12.08	78.24	0.00	1.37
BLASR	PacBio	INV	5000	29.54	53.29	12.83	0.00	4.29
BLASR	PacBio	INV	10000	31.99	59.28	2.38	0.05	5.76
BLASR	PacBio	INV	50000	36.63	61.10	0.94	0.00	0.98
BWA-mem	PacBio	DEL,INS	100	62.51	23.10	10.63	0.00	2.01
BWA-mem	PacBio	DEL,INS	250	25.65	22.97	41.07	0.42	10.65
BWA-mem	PacBio	DEL,INS	500	19.02	31.36	36.53	0.25	11.09
BWA-mem	PacBio	DEL,INS	1000	18.09	40.58	24.32	0.15	11.40
BWA-mem	PacBio	DEL,INS	5000	12.82	49.67	3.49	0.12	22.03
BWA-mem	PacBio	DEL,INS	10000	17.16	59.03	1.51	0.07	20.11
BWA-mem	PacBio	DEL,INS	50000	13.47	58.18	0.00	0.06	27.58
BWA-mem	PacBio	DUP	100	4.52	74.17	16.78	0.17	4.17
BWA-mem	PacBio	DUP	250	23.69	46.39	24.84	0.36	6.86
BWA-mem	PacBio	DUP	500	26.06	41.53	21.29	0.26	9.12
BWA-mem	PacBio	DUP	1000	27.13	54.18	12.49	0.43	6.63
BWA-mem	PacBio	DUP	5000	23.58	61.31	0.24	0.16	5.06
BWA-mem	PacBio	DUP	10000	24.27	71.56	0.09	0.18	1.60
BWA-mem	PacBio	DUP	50000	23.29	70.95	0.27	0.35	0.00
BWA-mem	PacBio	TRA	100	1.03	2.75	64.55	0.13	30.73
BWA-mem	PacBio	TRA	250	5.59	33.59	14.62	0.21	45.67
BWA-mem	PacBio	TRA	500	15.57	53.08	3.86	0.27	27.26
BWA-mem	PacBio	TRA	1000	22.91	57.14	2.13	0.22	17.24
BWA-mem	PacBio	TRA	5000	69.04	24.45	0.18	0.18	6.24
BWA-mem	PacBio	TRA	10000	82.11	10.09	0.02	0.14	7.27
BWA-mem	PacBio	TRA	50000	85.88	5.73	0.13	0.09	7.85
BWA-mem	PacBio	INV	100	2.27	2.43	64.99	0.41	29.90
BWA-mem	PacBio	INV	250	10.07	17.15	15.96	6.31	50.94
BWA-mem	PacBio	INV	500	18.28	36.95	3.19	15.40	25.54
BWA-mem	PacBio	INV	1000	23.93	53.58	4.27	5.50	12.44
BWA-mem	PacBio	INV	5000	24.37	68.91	0.16	0.47	6.16
BWA-mem	PacBio	INV	10000	28.93	64.76	0.00	0.32	5.58
BWA-mem	PacBio	INV	50000	25.37	64.78	0.00	0.21	9.24
GraphMap	PacBio	DEL,INS	100	37.47	29.62	26.58	0.00	0.00
GraphMap	PacBio	DEL,INS	250	35.71	40.86	18.28	0.00	0.00
GraphMap	PacBio	DEL,INS	500	38.40	43.09	7.47	0.00	0.00

GraphMap	PacBio	DEL,INS	1000	37.56	46.85	8.32	0.00	0.00
GraphMap	PacBio	DEL,INS	5000	9.15	15.99	54.13	0.00	0.20
GraphMap	PacBio	DEL,INS	10000	1.26	3.21	68.25	0.00	0.28
GraphMap	PacBio	DEL,INS	50000	0.00	0.13	84.35	0.00	1.73
GraphMap	PacBio	DUP	100	0.00	43.63	53.95	0.00	0.00
GraphMap	PacBio	DUP	250	0.00	68.66	28.05	0.00	0.00
GraphMap	PacBio	DUP	500	0.00	77.77	19.07	0.00	0.00
GraphMap	PacBio	DUP	1000	0.00	72.63	22.54	0.00	0.17
GraphMap	PacBio	DUP	5000	0.00	7.46	49.20	0.00	1.26
GraphMap	PacBio	DUP	10000	0.00	0.00	15.90	0.00	0.00
GraphMap	PacBio	DUP	50000	0.00	0.00	0.17	0.00	0.00
GraphMap	PacBio	TRA	100	0.00	0.00	97.17	0.00	0.00
GraphMap	PacBio	TRA	250	0.00	0.00	96.63	0.00	0.00
GraphMap	PacBio	TRA	500	0.00	0.00	96.84	0.00	0.08
GraphMap	PacBio	TRA	1000	0.00	0.00	92.23	0.00	1.59
GraphMap	PacBio	TRA	5000	0.00	0.00	78.78	0.00	4.47
GraphMap	PacBio	TRA	10000	0.00	0.00	57.93	0.00	20.34
GraphMap	PacBio	TRA	50000	0.00	0.00	58.05	0.00	22.70
GraphMap	PacBio	INV	100	0.00	0.00	98.46	0.00	0.16
GraphMap	PacBio	INV	250	0.00	0.00	98.55	0.00	0.17
GraphMap	PacBio	INV	500	0.00	0.00	97.05	0.00	0.32
GraphMap	PacBio	INV	1000	0.00	0.00	90.53	0.00	0.58
GraphMap	PacBio	INV	5000	0.00	0.00	80.39	0.00	0.52
GraphMap	PacBio	INV	10000	0.00	0.00	62.29	0.00	15.45
GraphMap	PacBio	INV	50000	0.00	0.00	61.49	0.00	18.40
NGMLR	PacBio	DEL,INS	100	93.92	2.95	0.17	0.00	0.00
NGMLR	PacBio	DEL,INS	250	90.56	7.98	0.17	0.00	0.17
NGMLR	PacBio	DEL,INS	500	85.49	10.95	0.43	0.00	1.39
NGMLR	PacBio	DEL,INS	1000	81.34	15.99	0.08	0.00	0.97
NGMLR	PacBio	DEL,INS	5000	51.29	42.62	0.07	0.14	0.81
NGMLR	PacBio	DEL,INS	10000	52.20	34.68	0.07	0.07	3.63
NGMLR	PacBio	DEL,INS	50000	40.64	44.47	0.00	0.06	0.77
NGMLR	PacBio	DUP	100	7.03	91.15	0.35	0.00	0.35
NGMLR	PacBio	DUP	250	87.62	9.70	0.09	0.17	1.21
NGMLR	PacBio	DUP	500	91.48	2.72	3.25	0.44	0.88
NGMLR	PacBio	DUP	1000	93.31	3.22	0.59	0.42	0.93
NGMLR	PacBio	DUP	5000	86.59	1.93	0.00	0.00	1.51
NGMLR	PacBio	DUP	10000	91.62	5.13	0.00	0.09	0.60

NGMLR	PacBio	DUP	50000	92.40	4.22	0.00	0.25	0.00
NGMLR	PacBio	TRA	100	23.99	1.07	13.86	0.09	59.40
NGMLR	PacBio	TRA	250	68.75	3.20	6.41	0.16	19.71
NGMLR	PacBio	TRA	500	83.81	5.78	3.08	0.27	5.05
NGMLR	PacBio	TRA	1000	91.15	6.00	0.11	0.29	1.30
NGMLR	PacBio	TRA	5000	95.47	2.24	0.03	0.16	1.34
NGMLR	PacBio	TRA	10000	96.04	0.51	0.00	0.16	2.04
NGMLR	PacBio	TRA	50000	96.71	0.09	0.00	0.18	1.95
NGMLR	PacBio	INV	100	87.20	6.65	4.29	0.24	0.81
NGMLR	PacBio	INV	250	87.37	7.42	2.65	0.17	1.37
NGMLR	PacBio	INV	500	91.94	2.55	0.88	0.80	1.84
NGMLR	PacBio	INV	1000	92.99	4.19	0.14	0.51	1.30
NGMLR	PacBio	INV	5000	96.07	1.71	0.00	0.36	1.09
NGMLR	PacBio	INV	10000	95.29	1.87	0.00	0.14	1.65
NGMLR	PacBio	INV	50000	96.75	1.37	0.00	0.26	0.56
Minimap2	PacBio	DEL,INS	100	98.23	0.34	0.00	0.00	0.00
Minimap2	PacBio	DEL,INS	250	98.45	0.43	0.17	0.00	0.00
Minimap2	PacBio	DEL,INS	500	88.88	8.17	0.17	0.00	0.26
Minimap2	PacBio	DEL,INS	1000	61.87	15.67	11.47	0.00	10.99
Minimap2	PacBio	DEL,INS	5000	51.08	45.93	0.14	0.00	1.69
Minimap2	PacBio	DEL,INS	10000	55.83	41.52	0.14	0.07	2.09
Minimap2	PacBio	DEL,INS	50000	40.77	55.72	0.00	0.06	1.85
Minimap2	PacBio	DUP	100	0.00	98.70	0.17	0.00	0.00
Minimap2	PacBio	DUP	250	0.00	97.49	0.52	0.00	0.61
Minimap2	PacBio	DUP	500	2.28	69.33	2.55	0.00	15.99
Minimap2	PacBio	DUP	1000	90.51	5.34	0.08	0.34	2.63
Minimap2	PacBio	DUP	5000	91.11	4.78	0.25	0.08	1.26
Minimap2	PacBio	DUP	10000	92.56	4.62	0.09	0.09	0.43
Minimap2	PacBio	DUP	50000	93.33	3.38	0.17	0.17	0.00
Minimap2	PacBio	TRA	100	0.00	0.00	98.50	0.00	0.00
Minimap2	PacBio	TRA	250	0.08	0.00	98.23	0.00	0.33
Minimap2	PacBio	TRA	500	4.51	0.12	0.35	0.08	94.37
Minimap2	PacBio	TRA	1000	18.90	0.00	0.11	0.07	79.44
Minimap2	PacBio	TRA	5000	95.53	1.74	0.00	0.13	2.55
Minimap2	PacBio	TRA	10000	96.20	0.39	0.00	0.14	2.94
Minimap2	PacBio	TRA	50000	96.64	0.00	0.00	0.15	2.70
Minimap2	PacBio	INV	100	0.00	0.00	98.78	0.00	0.16
Minimap2	PacBio	INV	250	0.00	0.00	98.63	0.00	0.77

Minimap2	PacBio	INV	500	4.79	0.32	13.17	0.24	80.29
Minimap2	PacBio	INV	1000	17.14	0.58	0.00	0.14	79.68
Minimap2	PacBio	INV	5000	93.79	3.26	0.00	0.16	2.69
Minimap2	PacBio	INV	10000	93.46	3.20	0.00	0.14	2.70
Minimap2	PacBio	INV	50000	91.83	5.39	0.00	0.17	2.01
LAST	PacBio	DEL,INS	100	48.35	14.18	22.19	14.09	1.77
LAST	PacBio	DEL,INS	250	55.45	22.15	8.76	12.70	5.67
LAST	PacBio	DEL,INS	500	53.43	28.67	3.48	12.68	6.95
LAST	PacBio	DEL,INS	1000	53.80	31.58	1.94	11.15	2.91
LAST	PacBio	DEL,INS	5000	33.13	48.64	3.12	6.37	8.06
LAST	PacBio	DEL,INS	10000	35.80	46.27	3.14	7.12	7.47
LAST	PacBio	DEL,INS	50000	26.26	52.59	4.35	6.45	9.84
LAST	PacBio	DUP	100	57.76	22.98	3.04	15.52	1.04
LAST	PacBio	DUP	250	56.97	23.38	3.64	15.84	0.43
LAST	PacBio	DUP	500	54.48	20.91	4.22	22.50	0.79
LAST	PacBio	DUP	1000	50.59	23.39	5.59	25.17	0.42
LAST	PacBio	DUP	5000	45.93	22.13	20.37	32.44	0.50
LAST	PacBio	DUP	10000	53.16	27.44	25.13	20.85	0.68
LAST	PacBio	DUP	50000	58.11	27.36	25.08	16.55	0.00
LAST	PacBio	TRA	100	30.13	27.73	0.09	10.52	31.03
LAST	PacBio	TRA	250	40.94	40.29	0.08	13.14	5.22
LAST	PacBio	TRA	500	46.49	37.01	0.04	15.42	1.16
LAST	PacBio	TRA	1000	51.64	32.96	0.04	13.84	1.01
LAST	PacBio	TRA	5000	70.94	19.08	0.11	9.03	0.97
LAST	PacBio	TRA	10000	72.69	22.68	0.12	3.22	1.06
LAST	PacBio	TRA	50000	74.14	23.23	0.07	1.28	0.95
LAST	PacBio	INV	100	47.33	16.29	0.00	16.69	20.18
LAST	PacBio	INV	250	50.68	23.29	0.00	18.52	7.85
LAST	PacBio	INV	500	53.55	26.02	0.16	18.68	0.96
LAST	PacBio	INV	1000	53.58	24.37	0.29	20.97	0.58
LAST	PacBio	INV	5000	53.65	26.13	0.00	19.35	0.93
LAST	PacBio	INV	10000	58.32	24.41	0.00	15.81	1.01
LAST	PacBio	INV	50000	59.91	25.67	0.00	13.18	0.73
MECAT	PacBio	DEL,INS	100	0.00	2.03	96.88	0.00	0.17
MECAT	PacBio	DEL,INS	250	0.00	0.09	99.66	0.00	4.21
MECAT	PacBio	DEL,INS	500	0.00	0.70	97.91	0.00	0.00
MECAT	PacBio	DEL,INS	1000	NaN	NaN	NaN	NaN	NaN
MECAT	PacBio	DEL,INS	5000	NaN	NaN	NaN	NaN	NaN

MECAT	PacBio	DEL,INS	10000	NaN	NaN	NaN	NaN	NaN
MECAT	PacBio	DEL,INS	50000	NaN	NaN	NaN	NaN	NaN
MECAT	PacBio	DUP	100	0.00	0.00	103.30	0.00	1.04
MECAT	PacBio	DUP	250	0.00	0.00	103.64	0.00	7.79
MECAT	PacBio	DUP	500	0.00	0.00	128.03	0.00	2.46
MECAT	PacBio	DUP	1000	0.00	0.08	120.85	0.00	2.37
MECAT	PacBio	DUP	5000	0.00	51.05	50.71	0.00	3.27
MECAT	PacBio	DUP	10000	0.09	72.82	15.98	0.00	0.43
MECAT	PacBio	DUP	50000	0.00	82.85	0.00	0.00	0.00
MECAT	PacBio	TRA	100	0.00	0.00	98.54	0.00	0.00
MECAT	PacBio	TRA	250	0.00	0.00	91.62	0.00	3.29
MECAT	PacBio	TRA	500	0.00	0.00	93.45	0.04	1.62
MECAT	PacBio	TRA	1000	0.47	0.98	92.41	0.00	5.46
MECAT	PacBio	TRA	5000	64.91	22.53	1.08	0.00	15.27
MECAT	PacBio	TRA	10000	82.02	4.28	8.29	0.07	16.62
MECAT	PacBio	TRA	50000	86.80	0.11	13.68	0.00	16.25
MECAT	PacBio	INV	100	0.00	0.00	99.11	0.00	0.00
MECAT	PacBio	INV	250	0.00	0.00	69.03	0.00	26.71
MECAT	PacBio	INV	500	0.00	0.00	42.78	12.93	39.90
MECAT	PacBio	INV	1000	0.00	0.00	98.41	0.22	1.23
MECAT	PacBio	INV	5000	0.00	87.64	19.30	0.00	13.55
MECAT	PacBio	INV	10000	0.00	82.91	3.47	0.00	17.50
MECAT	PacBio	INV	50000	0.00	83.57	0.00	3.17	12.71
BWA-mem	ONT	DEL,INS	100	61.46	20.78	12.94	0.00	2.20
BWA-mem	ONT	DEL,INS	250	20.37	22.59	47.03	0.07	9.86
BWA-mem	ONT	DEL,INS	500	19.24	33.93	33.21	0.00	13.19
BWA-mem	ONT	DEL,INS	1000	16.20	42.45	23.75	0.26	9.67
BWA-mem	ONT	DEL,INS	5000	10.97	53.75	6.48	0.05	20.88
BWA-mem	ONT	DEL,INS	10000	16.97	55.00	1.89	0.00	20.70
BWA-mem	ONT	DEL,INS	50000	10.73	58.46	0.00	0.05	27.08
BWA-mem	ONT	DUP	100	0.37	75.15	18.44	0.00	1.03
BWA-mem	ONT	DUP	250	14.76	50.30	27.82	0.15	4.75
BWA-mem	ONT	DUP	500	24.34	41.84	27.13	0.29	5.15
BWA-mem	ONT	DUP	1000	25.96	55.91	10.91	0.38	4.69
BWA-mem	ONT	DUP	5000	23.74	71.38	1.24	0.31	5.96
BWA-mem	ONT	DUP	10000	21.30	63.84	0.58	0.36	1.67
BWA-mem	ONT	DUP	50000	22.15	67.55	0.23	0.08	0.00
BWA-mem	ONT	TRA	100	0.74	3.54	70.72	0.07	23.13

BWA-mem	ONT	TRA	250	4.70	29.50	27.62	0.14	35.81
BWA-mem	ONT	TRA	500	13.54	52.80	7.50	0.24	24.49
BWA-mem	ONT	TRA	1000	24.05	56.33	2.02	0.19	15.67
BWA-mem	ONT	TRA	5000	68.06	24.19	0.17	0.27	6.04
BWA-mem	ONT	TRA	10000	83.65	8.39	0.02	0.06	6.23
BWA-mem	ONT	TRA	50000	89.90	0.98	0.04	0.15	6.93
BWA-mem	ONT	INV	100	2.47	2.47	69.75	0.14	22.97
BWA-mem	ONT	INV	250	8.59	17.03	27.06	5.77	39.83
BWA-mem	ONT	INV	500	18.18	38.24	5.69	15.82	19.99
BWA-mem	ONT	INV	1000	20.58	57.18	3.05	5.08	12.83
BWA-mem	ONT	INV	5000	22.82	68.88	0.55	0.64	5.83
BWA-mem	ONT	INV	10000	26.94	67.16	0.00	0.40	4.24
BWA-mem	ONT	INV	50000	24.37	66.06	0.00	0.19	7.40
GraphMap	ONT	DEL,INS	100	37.33	29.36	28.67	0.00	0.00
GraphMap	ONT	DEL,INS	250	36.84	34.37	24.00	0.00	0.00
GraphMap	ONT	DEL,INS	500	34.29	38.57	13.94	0.00	0.00
GraphMap	ONT	DEL,INS	1000	36.00	40.64	14.82	0.00	0.27
GraphMap	ONT	DEL,INS	5000	14.59	17.15	50.88	0.00	0.34
GraphMap	ONT	DEL,INS	10000	4.17	4.79	63.37	0.00	0.49
GraphMap	ONT	DEL,INS	50000	0.00	0.16	83.03	0.00	2.04
GraphMap	ONT	DUP	100	0.00	14.28	83.51	0.00	0.00
GraphMap	ONT	DUP	250	0.00	34.58	63.15	0.00	0.00
GraphMap	ONT	DUP	500	0.00	54.33	42.61	0.00	0.07
GraphMap	ONT	DUP	1000	0.00	62.63	31.66	0.00	0.15
GraphMap	ONT	DUP	5000	0.00	13.59	46.01	0.00	0.72
GraphMap	ONT	DUP	10000	0.00	1.69	23.81	0.00	0.39
GraphMap	ONT	DUP	50000	0.00	0.00	0.08	0.00	0.00
GraphMap	ONT	TRA	100	0.00	0.00	96.18	0.00	0.00
GraphMap	ONT	TRA	250	0.00	0.00	96.02	0.00	0.00
GraphMap	ONT	TRA	500	0.00	0.00	95.09	0.00	0.00
GraphMap	ONT	TRA	1000	0.00	0.00	90.11	0.00	2.50
GraphMap	ONT	TRA	5000	0.00	0.00	76.65	0.00	5.50
GraphMap	ONT	TRA	10000	0.00	0.00	54.89	0.00	21.69
GraphMap	ONT	TRA	50000	0.00	0.00	55.40	0.00	24.51
GraphMap	ONT	INV	100	0.00	0.00	97.17	0.00	0.28
GraphMap	ONT	INV	250	0.00	0.00	97.98	0.00	0.14
GraphMap	ONT	INV	500	0.00	0.00	95.84	0.00	0.07
GraphMap	ONT	INV	1000	0.00	0.00	91.99	0.00	0.83

GraphMap	ONT	INV	5000	0.00	0.00	78.86	0.00	1.23
GraphMap	ONT	INV	10000	0.00	0.00	58.12	0.00	18.42
GraphMap	ONT	INV	50000	0.00	0.00	57.67	0.00	19.36
NGMLR	ONT	DEL,INS	100	90.65	5.47	0.69	0.00	0.21
NGMLR	ONT	DEL,INS	250	87.16	7.69	0.07	0.00	0.51
NGMLR	ONT	DEL,INS	500	79.28	13.57	0.37	0.00	2.65
NGMLR	ONT	DEL,INS	1000	77.60	17.42	0.27	0.07	1.37
NGMLR	ONT	DEL,INS	5000	51.62	39.60	0.06	0.06	1.71
NGMLR	ONT	DEL,INS	10000	48.77	36.87	0.06	0.00	4.79
NGMLR	ONT	DEL,INS	50000	38.83	43.50	0.00	0.11	1.93
NGMLR	ONT	DUP	100	0.52	97.78	0.59	0.00	0.07
NGMLR	ONT	DUP	250	28.72	68.57	0.22	0.00	0.88
NGMLR	ONT	DUP	500	83.25	10.63	2.33	0.51	1.02
NGMLR	ONT	DUP	1000	89.37	6.47	0.77	0.39	0.39
NGMLR	ONT	DUP	5000	80.09	6.11	0.43	0.50	2.23
NGMLR	ONT	DUP	10000	86.44	7.94	0.00	0.69	0.31
NGMLR	ONT	DUP	50000	84.61	9.19	0.00	0.00	0.00
NGMLR	ONT	TRA	100	18.49	1.77	14.11	0.14	63.08
NGMLR	ONT	TRA	250	52.56	6.38	6.66	0.24	29.98
NGMLR	ONT	TRA	500	69.24	11.26	4.43	0.24	11.32
NGMLR	ONT	TRA	1000	80.86	10.28	0.35	0.26	4.14
NGMLR	ONT	TRA	5000	90.51	4.25	0.12	0.27	3.04
NGMLR	ONT	TRA	10000	91.34	2.42	0.00	0.24	3.95
NGMLR	ONT	TRA	50000	93.20	0.55	0.02	0.17	3.65
NGMLR	ONT	INV	100	82.76	6.78	3.04	0.99	3.46
NGMLR	ONT	INV	250	81.75	8.01	1.44	0.36	4.62
NGMLR	ONT	INV	500	83.97	5.90	0.90	0.76	4.09
NGMLR	ONT	INV	1000	85.01	8.32	0.13	0.25	2.16
NGMLR	ONT	INV	5000	90.98	4.60	0.05	0.50	2.41
NGMLR	ONT	INV	10000	89.46	5.39	0.00	0.59	2.73
NGMLR	ONT	INV	50000	90.85	5.32	0.04	0.27	1.29
LAST	ONT	DEL,INS	100	47.71	12.47	19.46	18.01	2.22
LAST	ONT	DEL,INS	250	49.17	23.21	7.25	16.82	6.02
LAST	ONT	DEL,INS	500	47.42	31.05	2.88	14.53	9.51
LAST	ONT	DEL,INS	1000	43.10	33.33	1.91	17.35	3.35
LAST	ONT	DEL,INS	5000	29.74	45.70	4.05	9.97	8.66
LAST	ONT	DEL,INS	10000	31.72	43.74	3.80	9.82	9.08
LAST	ONT	DEL,INS	50000	21.54	53.92	4.51	8.11	10.74

LAST	ONT	DUP	100	48.89	27.66	4.36	18.05	0.07
LAST	ONT	DUP	250	45.27	26.45	4.47	23.08	0.07
LAST	ONT	DUP	500	45.81	23.02	5.46	26.73	0.00
LAST	ONT	DUP	1000	45.22	21.96	7.63	31.43	0.00
LAST	ONT	DUP	5000	36.09	19.19	19.55	44.00	0.00
LAST	ONT	DUP	10000	39.68	23.11	30.35	37.29	0.00
LAST	ONT	DUP	50000	50.61	27.79	32.77	21.52	0.00
LAST	ONT	TRA	100	25.46	27.97	0.07	12.41	32.36
LAST	ONT	TRA	250	33.69	42.81	0.03	17.15	4.29
LAST	ONT	TRA	500	38.92	39.29	0.00	19.20	1.40
LAST	ONT	TRA	1000	44.09	34.81	0.19	17.69	1.61
LAST	ONT	TRA	5000	64.65	18.30	0.05	14.12	1.71
LAST	ONT	TRA	10000	64.85	24.40	0.06	7.68	1.69
LAST	ONT	TRA	50000	63.10	32.06	0.06	1.58	1.47
LAST	ONT	INV	100	40.14	19.36	0.07	17.46	20.92
LAST	ONT	INV	250	47.76	24.60	0.00	19.91	6.13
LAST	ONT	INV	500	48.51	25.82	0.07	22.21	1.73
LAST	ONT	INV	1000	45.43	30.30	0.13	21.54	1.40
LAST	ONT	INV	5000	44.56	27.11	0.00	25.97	1.32
LAST	ONT	INV	10000	48.26	25.32	0.00	23.53	1.82
LAST	ONT	INV	50000	51.78	28.21	0.00	17.31	0.99

Table B.1: Full evaluation results for simulated SVs and simulated reads.

Program	Techn.	Type	Length	Precise	Indicated	Incorrect	Fragmented	Not found
BLASR	PacBio	DEL	1000	6.92	71.14	9.57	0.00	9.78
BLASR	PacBio	DEL	2000	5.84	47.19	4.25	0.00	36.34
BLASR	PacBio	DEL	3000	6.52	34.26	4.61	0.00	48.24
BLASR	PacBio	INS	1000	5.84	35.89	28.55	0.00	23.32
BLASR	PacBio	INS	2000	6.25	21.62	3.75	0.00	45.93
BLASR	PacBio	INS	3000	2.45	9.47	2.23	0.00	50.65
BLASR	PacBio	TRA	1000	NaN	NaN	NaN	NaN	NaN
BLASR	PacBio	TRA	2000	36.82	10.58	41.39	0.70	7.40
BLASR	PacBio	TRA	3000	51.91	14.55	19.33	0.69	7.69
BLASR	PacBio	INV	1000	2.89	5.63	84.24	0.00	3.97
BLASR	PacBio	INV	2000	14.71	30.44	44.58	0.00	7.73
BLASR	PacBio	INV	3000	19.30	44.69	23.84	0.08	7.50
BWAMEM	PacBio	DEL	1000	40.17	30.60	20.98	1.53	5.23

BWAMEM	PacBio	DEL	2000	53.64	28.91	4.40	2.28	4.63
BWAMEM	PacBio	DEL	3000	48.83	39.17	2.34	1.83	4.39
BWAMEM	PacBio	INS	1000	11.26	11.92	52.20	1.36	16.92
BWAMEM	PacBio	INS	2000	6.45	10.01	36.55	1.02	25.45
BWAMEM	PacBio	INS	3000	4.72	5.61	25.47	1.08	32.08
BWAMEM	PacBio	TRA	1000	NaN	NaN	NaN	NaN	NaN
BWAMEM	PacBio	TRA	2000	47.55	26.72	15.53	2.00	6.22
BWAMEM	PacBio	TRA	3000	61.64	24.30	1.97	2.33	4.95
BWAMEM	PacBio	INV	1000	7.07	5.89	77.22	1.77	4.82
BWAMEM	PacBio	INV	2000	34.89	24.62	26.71	5.38	6.49
BWAMEM	PacBio	INV	3000	39.66	38.75	8.28	5.08	4.67
GraphMap	PacBio	DEL	1000	19.40	59.04	19.24	0.00	0.05
GraphMap	PacBio	DEL	2000	19.65	55.69	10.09	0.00	0.46
GraphMap	PacBio	DEL	3000	11.27	49.19	16.91	0.00	0.29
GraphMap	PacBio	INS	1000	34.95	17.57	38.04	0.00	0.84
GraphMap	PacBio	INS	2000	19.43	12.74	36.24	0.00	1.25
GraphMap	PacBio	INS	3000	9.80	5.46	39.84	0.00	2.27
GraphMap	PacBio	TRA	1000	NaN	NaN	NaN	NaN	NaN
GraphMap	PacBio	TRA	2000	0.00	0.00	91.15	0.00	2.72
GraphMap	PacBio	TRA	3000	0.00	0.00	85.39	0.00	4.05
GraphMap	PacBio	INV	1000	0.00	0.00	94.91	0.00	0.48
GraphMap	PacBio	INV	2000	0.00	0.00	92.53	0.00	1.87
GraphMap	PacBio	INV	3000	0.00	0.00	90.21	0.00	1.43
NGMLR	PacBio	DEL	1000	77.06	8.77	2.64	0.11	7.56
NGMLR	PacBio	DEL	2000	69.80	8.57	1.06	1.06	13.43
NGMLR	PacBio	DEL	3000	70.06	9.66	0.66	1.17	12.15
NGMLR	PacBio	INS	1000	47.52	12.62	5.37	0.33	25.98
NGMLR	PacBio	INS	2000	24.24	6.92	2.54	0.90	39.91
NGMLR	PacBio	INS	3000	13.44	4.49	2.12	0.97	43.15
NGMLR	PacBio	TRA	1000	NaN	NaN	NaN	NaN	NaN
NGMLR	PacBio	TRA	2000	68.70	15.96	2.17	1.17	6.58
NGMLR	PacBio	TRA	3000	68.82	14.68	1.82	1.06	6.89
NGMLR	PacBio	INV	1000	58.04	8.20	12.00	2.89	9.11
NGMLR	PacBio	INV	2000	73.82	10.44	1.78	3.91	5.78
NGMLR	PacBio	INV	3000	73.66	11.92	0.98	3.32	5.12
BWAMEM	ONT	DEL	1000	50.69	31.78	13.87	0.38	2.13
BWAMEM	ONT	DEL	2000	46.18	41.99	3.35	1.47	3.98
BWAMEM	ONT	DEL	3000	40.63	48.39	3.83	0.91	3.02

BWAMEM	ONT	INS	1000	5.87	10.12	69.47	0.23	9.58
BWAMEM	ONT	INS	2000	4.42	14.86	46.92	1.27	19.81
BWAMEM	ONT	INS	3000	4.84	13.91	34.50	0.95	24.60
BWAMEM	ONT	TRA	1000	NaN	NaN	NaN	NaN	NaN
BWAMEM	ONT	TRA	2000	33.39	35.36	22.88	1.39	4.51
BWAMEM	ONT	TRA	3000	47.65	38.05	3.32	2.06	4.07
BWAMEM	ONT	INV	1000	5.35	4.79	83.96	1.84	2.55
BWAMEM	ONT	INV	2000	27.50	25.65	30.02	5.63	5.63
BWAMEM	ONT	INV	3000	34.29	43.20	8.56	7.01	3.51
GraphMap	ONT	DEL	1000	19.89	59.53	19.21	0.00	0.00
GraphMap	ONT	DEL	2000	19.16	67.64	7.54	0.00	0.10
GraphMap	ONT	DEL	3000	14.01	61.90	12.80	0.00	0.10
GraphMap	ONT	INS	1000	16.23	19.63	58.89	0.00	0.15
GraphMap	ONT	INS	2000	16.20	23.29	43.51	0.00	0.54
GraphMap	ONT	INS	3000	10.41	20.20	40.40	0.00	1.06
GraphMap	ONT	TRA	1000	NaN	NaN	NaN	NaN	NaN
GraphMap	ONT	TRA	2000	0.00	0.00	95.28	0.00	0.88
GraphMap	ONT	TRA	3000	0.00	0.00	90.66	0.00	1.65
GraphMap	ONT	INV	1000	0.00	0.00	98.00	0.00	0.32
GraphMap	ONT	INV	2000	0.00	0.00	92.22	0.00	0.30
GraphMap	ONT	INV	3000	0.00	0.00	92.71	0.00	0.28
NGMLR	ONT	DEL	1000	84.76	8.38	0.76	0.08	1.98
NGMLR	ONT	DEL	2000	71.41	16.75	0.94	0.21	5.13
NGMLR	ONT	DEL	3000	65.63	18.15	0.40	0.20	6.55
NGMLR	ONT	INS	1000	43.04	27.82	9.27	0.00	11.90
NGMLR	ONT	INS	2000	25.77	24.36	5.76	0.54	26.04
NGMLR	ONT	INS	3000	17.81	19.76	4.12	0.45	31.89
NGMLR	ONT	TRA	1000	NaN	NaN	NaN	NaN	NaN
NGMLR	ONT	TRA	2000	52.13	29.18	9.99	0.45	4.06
NGMLR	ONT	TRA	3000	57.03	24.98	6.81	0.68	4.10
NGMLR	ONT	INV	1000	36.95	18.44	23.70	0.96	15.00
NGMLR	ONT	INV	2000	55.30	29.50	2.82	1.85	2.97
NGMLR	ONT	INV	3000	57.36	31.63	0.77	1.40	3.37

Table B.3: Mapping comparison for simulated reference and real reads

Caller	Read length	Category	Indel	Dup	Inv	Tra	Invdel	Invdup
DELLY	short	correct	37.82	62.61	72.31	83.28	0.00	27.16
DELLY	short	indicated	0.00	4.81	26.33	14.61	69.62	52.45
DELLY	short	notfound	60.82	30.71	0.00	1.42	27.68	6.93
DELLY	short	incorrect	1.36	1.86	1.36	0.70	2.70	13.46
Lumpy	short	correct	52.14	89.29	93.57	96.07	2.38	30.58
Lumpy	short	indicated	0.00	5.00	6.43	3.57	64.39	34.80
Lumpy	short	notfound	47.86	5.71	0.00	0.36	25.97	6.49
Lumpy	short	incorrect	0.00	0.00	0.00	0.00	7.26	28.13
Manta	short	correct	2.73	11.39	25.27	49.01	0.23	5.15
Manta	short	indicated	33.12	49.60	62.99	46.10	64.74	73.63
Manta	short	notfound	57.66	26.84	0.00	0.31	25.52	5.32
Manta	short	incorrect	6.49	12.17	11.74	4.58	9.52	15.90
SURVIVOR (2+caller)	short	correct	2.86	12.76	18.57	41.07	0.24	5.23
SURVIVOR (2+caller)	short	indicated	35.71	55.85	81.43	58.57	67.95	74.54
SURVIVOR (2+caller)	short	notfound	61.43	30.71	0.00	0.36	27.85	6.93
SURVIVOR (2+caller)	short	incorrect	0.00	0.68	0.00	0.00	3.96	13.29
PBHoney	Pacbio	correct	26.73	0.00	59.29	44.29	0.00	5.32
PBHoney	Pacbio	indicated	25.06	82.86	9.29	45.71	19.68	34.00
PBHoney	Pacbio	notfound	42.14	17.14	31.43	10.00	80.08	44.26
PBHoney	Pacbio	incorrect	6.06	0.00	0.00	0.00	0.23	16.42
Sniffles(BWA)	Pacbio	correct	80.17	77.86	79.39	79.03	21.92	62.87
Sniffles(BWA)	Pacbio	indicated	12.07	19.29	5.00	8.85	41.71	23.40
Sniffles(BWA)	Pacbio	notfound	6.39	2.86	14.93	11.42	31.83	10.34
Sniffles(BWA)	Pacbio	incorrect	1.36	0.00	0.68	0.70	4.55	3.39
Sniffles(NGMLR)	Pacbio	correct	97.14	84.29	97.86	97.50	63.60	72.15
Sniffles(NGMLR)	Pacbio	indicated	0.71	14.29	2.14	1.79	27.01	18.93
Sniffles(NGMLR)	Pacbio	notfound	2.14	1.43	0.00	0.71	4.73	8.57
Sniffles(NGMLR)	Pacbio	incorrect	0.00	0.00	0.00	0.00	4.65	0.35
Sniffles+BWA	Nanopore	correct	81.29	70.84	79.75	73.52	30.89	61.67
Sniffles+BWA	Nanopore	indicated	6.65	21.74	4.08	19.86	50.06	26.39
Sniffles+BWA	Nanopore	notfound	5.44	2.04	10.17	4.18	12.65	7.60
Sniffles+BWA	Nanopore	incorrect	6.62	5.38	6.00	2.44	6.40	4.34
Sniffles+NGM	Nanopore	correct	87.37	62.04	87.05	92.56	63.21	71.47
Sniffles+NGM	Nanopore	indicated	0.00	22.16	1.33	2.03	26.26	17.56
Sniffles+NGM	Nanopore	notfound	1.95	4.49	0.00	0.68	1.41	5.29
Sniffles+NGM	Nanopore	incorrect	10.68	11.30	11.62	4.73	9.11	5.68

Table B.2: SVs caller statistics for simulated reads

Caller	Dataset	Read length	Category	# SVs	SVs(%)
DELLY	sim ref + real reads	short	Precise	551	78.71
DELLY	sim ref + real reads	short	Indicated	36	5.14
DELLY	sim ref + real reads	short	Not found	113	16.14
Lumpy	sim ref + real reads	short	Precise	538	76.86
Lumpy	sim ref + real reads	short	Indicated	49	7.00
Lumpy	sim ref + real reads	short	Notfound	113	16.14
Manta	sim ref + real reads	short	Precise	622	88.86
Manta	sim ref + real reads	short	Indicated	42	6.00
Manta	sim ref + real reads	short	Not found	36	5.14
SURVIVOR (2+caller)	sim ref + real reads	short	Precise	536	76.57
SURVIVOR (2+caller)	sim ref + real reads	short	Indicated	50	7.14
SURVIVOR (2+caller)	sim ref + real reads	short	Not found	114	16.29
Sniffles (NGMLR)	sim ref + real reads	Pacbio	Precise	666	95.14
Sniffles (NGMLR)	sim ref + real reads	Pacbio	Indicated	10	1.43
Sniffles (NGMLR)	sim ref + real reads	Pacbio	Not found	24	3.43
Sniffles (NGMLR)	sim ref + real reads	Nanopore	Precise	618	88.29
Sniffles (NGMLR)	sim ref + real reads	Nanopore	Indicated	49	7.00
Sniffles (NGMLR)	sim ref + real reads	Nanopore	Not found	33	4.71

Table B.4: SVs caller comparison for simulated reference and real reads.

Tool	DEL	DUP	INS	INV	TRA
Delly	13555	1864		3467	18103
Lumpy	11847	1008		1930	5760
Manta	12264	1567	3196	1181	10398
SURV	7936	1046	0	1238	6903
Sniffles	9806	1251	11300	876	1562

(a) Union (3 samples)

Tool	DEL	DUP	INS	INV	TRA
Delly	2922	439		469	1613
Lumpy	1317	375		609	0
Manta	3556	498	1094	376	1861
SURV	2414	353		394	801
Sniffles	3743	496	4226	67	24

(b) Cat1: found in all(%)

Tool	DEL	DUP	INS	INV	TRA
Delly	5337	820		856	3730
Lumpy	2482	628		1134	1913
Manta	6443	887	1814	683	4015
SURV	4224	621	0	720	2575
Sniffles	6798	828	7606	137	107

(c) Cat2: son, at least one parent (%)

Tool	DEL	DUP	INS	INV	TRA
Delly	1962	253		1042	5138
Lumpy	1109	65		134	1940
Manta	1328	108	288	80	1438
SURV	595	70		86	1533
Sniffles	726	114	1208	67	91

(d) Cat3: son, not in parents (%)

Tool	DEL	DUP	INS	INV	TRA
Delly	1962	253	0	1042	5138
Lumpy	1109	65	0	134	1940
Manta	1328	108	288	80	1438
SURV	677	75	0	90	1550
Sniffles	515	115	1045	66	90

(e) Cat3: son, not in parents accounted for length (%)

Length cuttoffs	50bp	200bp
#SVs supported by 1 out of 5 data sets	24392	9941
#SVs supported by 2 out of 5 data sets	4782	1929
#SVs supported by 3 out of 5 data sets	8289	3327
#SVs supported by 4 out of 5 data sets	2086	1377
#SVs supported by 5 out of 5 data sets	1052	827
Unique Pacbio	773	307
70% overlap repeats	323	74
Unique Nanopre	11433	1146
70% overlap repeats	10250	442
Unique illumina	4070	3479
70% overlap repeats	41	21
Unique GIAB Pacbio	7637	4711
70% overlap repeats	994	462
Unique 1k deletions	479	298
70% overlap repeats	13	0

Table B.6: Comparison of existing NA12878 data sets.

Program	Type	Tested	p<0.01	p<0.01 (%)
Delly	DEL	8842	2474	27.98
Lumpy	DEL	2387	1534	64.26
Manta	DEL	5301	2868	54.10
SURVIVOR	DEL	3102	1873	60.38
Sniffles + Pac	DEL	6399	3415	53.37
Sniffles + Nano	DEL	12045	3879	32.20
Manta	INS	1095	629	57.44
Sniffles + Pac	INS	5786	2685	46.41
Sniffles + Nano	INS	3488	1703	48.82

Table B.7: Indel assessment using Illumina short read data.

Illumina	Pacbio	Count	Percent
dup	INV	9	1.63
dup	INVDUP	7	1.27
dup	INV_INVDUP	0	0.00
dup	DUP	197	35.62
dup	DUP_INS	4	0.72
dup	INS	172	31.10
dup	DEL	21	3.80
dup	TRA	11	1.99
dup	highcov	35	6.33
dup	repeats	50	9.04
dup	no overlap	47	8.50
inv	INV	143	19.56
inv	INVDUP	10	1.37
inv	INV_INVDUP	0	0.00
inv	DUP	8	1.09
inv	DUP_INS	0	0.00
inv	INS	129	17.65
inv	DEL	32	4.38
inv	TRA	12	1.64
inv	highcov	24	3.28
inv	repeats	221	30.23
inv	no overlap	152	20.79
tra	TRA	295	13.13
tra	INS	1098	48.87
tra	DUP_INS	0	0.00
tra	DEL	199	8.86
tra	DUP	5	0.22
tra	INV	8	0.36
tra	INVDUP	14	0.62
tra	INV_INVDUP	0	0.00
tra	highcov	141	6.28
tra	repeats	404	17.98
tra	no overlap	83	3.69
del	TRA	55	1.47
del	INS	352	9.40
del	DUP_INS	0	0.00
del	DEL	2905	77.59
del	DUP	28	0.75
del	INV	1	0.03
del	INVDUP	3	0.08
del	INV_INVDUP	0	0.00
del	highcov	68	1.82
del	repeats	168	4.49
del	no overlap	164	4.38

Table B.8: Analysis of potential biases in short read calling.

Dataset	Program	Runtime [s]	Memory [MB]	Threads
NA12878 PacBio 1x	ngmlr-0.2.3	4,788	10,238	10
NA12878 PacBio 1x	bwa	6,133	6,039	10
NA12878 PacBio 1x	graphmap	557,493 (failed)	106,923	10
NA12878 PacBio 1x	blasr	18,518	19,337	10
NA12878 PacBio 1x	minimap2	546	10,051	10
NA12878 PacBio 1x	mecat	1,236	22,935	10
NA12878 PacBio 1x	lastal	5,575	17,367	10
NA12878 PacBio 1x	last-split	5,755	2,118	1
NA12878 PacBio 1x	last-mafconvert	991	10	1
NA12878 PacBio 1x	last sum	12,321	17,367	10
NA12878 PacBio full	sniffles-1.0.5	12,102	15,270	10
NA12878 ONT full	sniffles-1.0.5	7,744	5,238	10

Table B.9: Runtime comparisons for NA12878

Name	Technologie	Organism	Sample	Accession
Pac_skbr3	PacBio	human	SKBR3	https://www.biorxiv.org/content/early/2017/08/10/174938
Pac_giab	PacBio	human	H002	ftp://ftp-trace.ncbi.nlm.nih.gov/giab/ftp/data/AshkenazimTrio/
Pac_giab	PacBio	human	H003	ftp://ftp-trace.ncbi.nlm.nih.gov/giab/ftp/data/AshkenazimTrio/
Pac_giab	PacBio	human	H004	ftp://ftp-trace.ncbi.nlm.nih.gov/giab/ftp/data/AshkenazimTrio/
Nanopore_NA12878	ONT	human	NA12878	https://github.com/nanopore-wgs-consortium/NA12878
NA12878_PacBio_MtSinai	PacBio	human	NA12878	ftp://ftp-trace.ncbi.nlm.nih.gov/giab/ftp/data/NA12878/NA12878_PacBio_MtSinai
NA12878_giab_calls	Pacbio	human	NA12878	ftp://ftp-trace.ncbi.nih.gov/giab/ftp/data/NA12878/NA12878_PacBio_MtSinai/

Table B.10: Real data sets and accessions

Parameter	Default	Explanation
-r, –reference	NA	Path to the reference genome (FASTA/Q, can be gzipped)
-q, –query	NA	Path to the read file (FASTA/Q) [/dev/stdin]
-o, –output		Path to output file [stdout]
–skip-write	false	Don't write reference index to disk
–bam-fix	false	Report reads with >64k CIGAR operations as unmapped. Required to be compatibel to BAM format [false]
-t, –threads	1	Number of threads
-x, –presets	pacbio	Parameter presets for different sequencing technologies
-i, –min-identity	0.65	Alignments with an identity lower than this threshold will be discarded
-R, –min-residues	0.25	Alignments containing less than <int>or (<float> * read length) residues will be discarded
–no-smallinv	false	Don't detect small inversions
–no-lowqualitysplit	false	Split alignments with poor quality
–verbose	false	Debug output
–no-progress	false	Don't print progress info while mapping
–match	2	Match score
–mismatch	-5	Mismatch score
–gap-open	-5	Gap open score
–gap-extend-max	-5	Gap open extend max
–gap-extend-min	-1	Gap open extend min
–gap-decay	0.15	Gap extend decay
-k, –kmer-length	2	Number of k-mers to skip when building the lookup table from the reference
–bin-size	4	Sets the size of the grid used during candidate search
–max-segments	1	Max number of segments allowed for a read per kb
–subread-length	256	Length of fragments reads are split into
–subread-corridor	40	Length of corridor sub-segments are aligned with

Table B.11: NGMLR command line parameters.