Contributions to the detection of non-reference sequences in population-scale NGS data

Dissertation

zur Erlangung des Grades

eines Doktors der Naturwissenschaften (Dr. rer. nat.)

am Fachbereich für Mathematik und Informatik

der Freien Universität Berlin

vorgelegt von Thomas Krannich

Berlin 2022

Erstgutachter:	Prof. Dr. Knut Reinert
Zweitgutachterin:	Prof. Dr. Birte Kehr
Tag der Disputation:	Freitag, 29. April 2022

*"Finding your way in life is like finding the genome in a De Bruijn Graph: it is very easy to find *a* path, very hard to find *the* path."*

Lex Nederbragt

Zusammenfassung

Fehlende Sequenzen im Referenzgenom (englische Abk. *NRS*) sind eine vergleichsweise wenig untersuchte Klasse an genomischer Strukturvarianten. NRS sind Sequenzen im Genom einzelner Individuen, welche bei einem Vergleich mit einem gegebenen Referenzgenom neuartig erscheinen. Eine gängige These zum Ursprung der NRS ist, dass diese keine neuen Sequenzen in einem einzelnen Individuum darstellen, sondern vorrangig Indikatoren für fehlende genetische Diversität im Referenzgenom sind. Da Referenzgenome, wie zum Beispiel das menschliche Referenzgenom, überwiegend aus einzelnen Genomen erstellt wurden, weisen neue und nicht für das Referenzgenom genutzte Genome einzelner Individuen Sequenzen auf, welche folglich nicht im Referenzgenom auffindbar sind.

Im modernen Zeitalter der Hochdurchsatz-Sequenzierung lässt sich genomische Strukturvarianten routinemäßig in hunderten und tausenden an Individuen feststellen. Beim Auffinden von NRS mit Hilfe von kurzen Sequenzierungs-Daten ist jedoch eine Rekonstruktion längerer genomischer Sequenzen unerlässlich. Dieser Prozess der Rekonstruktion ist ein rechnerisch komplexes Problem und benötigt große Mengen hochqualitativer Daten. Studien haben gezeigt, dass das Vereinen von Sequenzierungs-Daten mehrerer Individuen eine zuverlässige Methode ist, um gemeinsame NRS innerhalb einer Studiengruppe oder Population aufzufinden. Die in diesen Studien benutzen Algorithmen zeigen jedoch eine begrenzte Anwendbarkeit für große Datensätze auf.

Im Rahmen dieser Doktorarbeit werden neue Konzepte vorgestellt, um NRS gleichzeitig in mehreren Individuen zu entdecken. Ein wichtiger Fokus hierbei ist, dass diese neuen Konzepte große Datensätze verarbeiten können und sich auf eine zuvor unerreichte Anzahl an individuellen Genomen anwenden lassen. Zudem wird eine Software vorgestellt, genannt *PopIns2*, mit welcher sich die vorgestellten Konzepte in der Praxis anwenden lassen. Den wichtigsten Beitrag leistet dabei ein neuer Algorithmus zur Vereinigung rekonstruierter genomischer Sequenzen aus mehreren Individuen. Die vereinte Menge an genomischen Sequenzen bildet anschließend die Grundlage zur präzisen Charakterisierung von NRS in den Genomen vieler Individuen. Versuche mit künstlichen Sequenzierungs-Daten haben gezeigt, dass PopIns2 und der darin enthaltene neue Algorithmus ein genaues Auffinden von NRS in großen Datensätzen ermöglicht. Zudem haben Versuche mit echten Sequenzierungs-Daten gezeigt, dass PopIns2 das Auffinden von NRS in Datensätzen ermöglicht, deren Größenordnung die rechnerische Leistungsgrenze vorhergehender Programme weit übersteigt.

Abstract

Non-reference sequence (NRS) variants are a less frequently investigated class of genomic structural variants (SV). Here, DNA sequences are found within an individual that are novel with respect to a given reference. NRS occur predominantly due to the fact that a linear reference genome lacks biological diversity and ancestral sequence if it was primarily derived from a single or few individuals. Therefore, newly sequenced individuals can yield genomic sequences which are absent from a reference genome.

With the increasing throughput of sequencing technologies, SV detection has become possible across tens of thousands of individuals. When using short-read data, the detection of NRS variants inevitably involves a de novo assembly which is a complex computational problem and requires high-quality sequence data at high coverage. Previous studies have demonstrated how sequence data of multiple genomes can be combined for the reliable detection of NRS variants. However, the algorithms proposed in these studies have a limited capability to process large sets of genomes.

This thesis introduces novel contributions for the discovery of NRS variants in many genomes, which scale to considerably larger numbers of genomes than previous methods. A practical software tool, *PopIns2*, that was developed to apply the presented methods is elucidated in greater detail. The highlight among the new contributions is a procedure to merge contig assemblies of unaligned reads from many individuals into a single set of NRS by heuristically generating a weighted minimum path cover for a colored de Bruijn graph. Tests on simulated data show that PopIns2 ranks among the best approaches in terms of quality and reliability and that its approach yields the best precision for a growing number of genomes processed. Results on the Polaris Diversity Cohort and a set of 1000 Icelandic human genomes demonstrate unmatched scalability for the application on population-scale datasets.

Acknowledgements

First and foremost, I would like to thank my supervisor Birte Kehr. I am ineffable grateful for all the guidance, support and opportunities I was given and provided throughout the years. During my initial month and years as PhD student I experienced an incomparable patience and dedication to introduce me into the field of sequence analysis and variant calling, particularly when I struggled and doubted my capabilities the most. Birte has always been available for advise and has consistently propelled my interest and enthusiasm for my work. I am very thankful for her kind and pleasant mentoring that taught me numerous skills and a vast amount of knowledge. Also, when I became more confident in my field of research, I was granted the trust to manage scientific work in a self-reliant manner. I am also particularly thankful for all the support and opportunities I was granted to publically share and present my research, gain scientific experience from industry leading companies and to expand my professional network across the globe. Birte's professionalism and passion for her work is unprecedented and has been a great encouragement for myself to follow her example.

I am grateful to Knut Reinert who has accompanied my way from the early days of my Bachelor studies until the final days of my PhD studies. I really enjoy the memories of his lectures on Algorithms and Data structures in Bioinformatics which have been, without a doubt, the major driver for me to carry on with the Bioinformatics major. During my PhD seminars he provided valuable feedback on my projects and his *SeqAn* team has provided a fundamental resource and support for my work on Bioinformatics software.

I would like to thank the all current and former members of the Kehr Lab. I consistently experienced a pleasant working atmosphere throughout my PhD years at the BIH and could always find a helping hand when needed. I appreciated our open and lively exchange of ideas and feedback in the office a lot. Many thanks to Sebastian Niehus and Timothy White for supporting and co-authoring my scientific work. Also, I am grateful for the support and many scientific discussions with the members of the Kircher Lab, Core Unit Bioinformatics (at the BIH) and Rayan Chikhi (at several international conferences).

I believe that getting oneself a PhD position is not just about a presentable skill set and the impression you leave in an interview but you also need a strong advocate. I would like to thank Annalisa Marsico for her flattering positive evaluation of my work at the MPIMG and for recommending me for my PhD position.

Last but not least, I am forever grateful for the steady support of my family.

Contents

List of Figures

List of Tables

1 Introduction

Since the dawn of civilisation people have tried to find the origin of life, understand the proliferation of physical traits and characterize the diversity of populations. A variety of species ranging from bacteria and fungi to birds, fish, reptiles and mammals all unite characteristics like growth, heredity, procreation and metabolism. Even if not understood in the past, and still not fully understood today, the hypothesis emerged [Sutton, 1903; Punnett, 1926; Portin, 2014] that all living organisms must underlay some common principles of life. Therefore, early on researchers attempted to find a blue print for the origin of life [Watson and Crick, 1953*a*; Gamow, 1954].

The first discovery that led to our understanding of genetics today was in 1869 when Friedrich Miescher accidentally discovered an unknown substance during his research on leucocytes [Miescher, 1871]. As he extracted the new substance from the nucleus he labelled it "Nuklein" (ger. "nuclein"). That discovery marks the birth of what we know today as *deoxyribonucleic acid* (DNA).

Hierarchies of DNA – From nucleotides to genomes

The four nucleobases, adenine (A), cytosine (C), guanine (G) and thymine (T), are the building blocks of DNA. Together with a five-carbon sugar and one or more phosphate groups they build nucleotides which are considered the smallest units of genetic information. In a living cell these four nucleotides are connected to *DNA strands* that chain up to millions of nucleotides with a covalent chemical bond. A more general term for a chain of nucleotides of arbitrary length is *DNA sequence* and is abstracted by solely its nucleobase abbreviations. In genetics the chemical term *base* is often used interchangeably for nucleotide, even if not chemically accurate.

After Miescher's discovery it took almost a century until Francis Crick and James Watson in 1953 first described the molecular structure of DNA [Watson and Crick, 1953*b*]. In their model, which in general is accepted till this day, they first described the *DNA double helix*. The DNA double helix is composed of two winding DNA strands (Figure 1). The

sugar-phosphate bonds of the nucleotides build the *backbone* of the helix while their nucleobases form the interconnection between the strands with non-covalent hydrogen bonds. Two interconnected bases of the two strands are called a *base pair*. Bases normally only pair in the combinations adenine-thymine and cytosine-guanine, i.e. if for instance an adenine base is observed at one DNA strand, the opposing strand contains a thymine as *complementary base*. The opposing strands in a DNA helix are paired in an anti-parallel direction. The *directionality* of a strand is determined by its endings. To put it simply, the strand ending with the phosphate group is called *5'-end* and the other end (of the same strand) is called *3'-end*. Because of these two properties (anti-directional strands and complementary base pairing) one strand of the DNA helix is said to be *reverse complement* to the other strand.

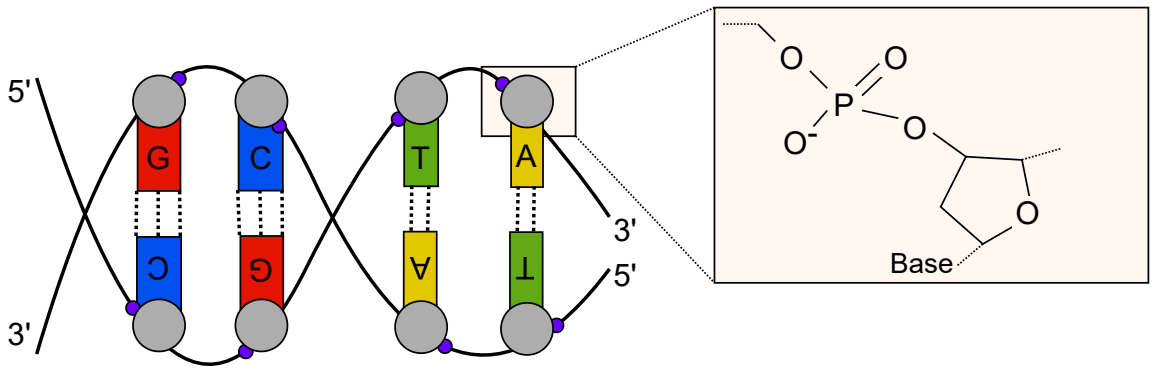

Figure 1: Excerpt of the DNA double helix. The two winding DNA strands (black) are held together by the hydrogen bridge bonds (dotted lines) of the base pairs (red-blue and green-yellow). The bases pair only in the specific combinations adenine-thymine (A and T) and cytosine-guanine (C and G). The DNA strands have a directionality (3'-end and 5'-end) defined by the tail that ends with a phosphate group (purple). One tail is called *5'-end* because the phosphate group binds to the fifth carbon atom in the pentose sugar ring (zoom box).

In a living cell, the DNA double helix would be too long and too vulnerable to structural damage to remain in its linearly outstretched form inside the nucleus. Therefore, cells usually keep the DNA double helix in a more compacted and shorter form. Inside the nucleus there exists a protein complex, called *histone*, to densely pack the DNA. The DNA winds around the histones in a highly ordered manner such that the 2 nm thin DNA double helix forms a 30 nm tube-like fiber. Subsequently, the 30 nm fiber forms loops and folds into an even wider and higher compacted fiber. The final tight coiling of the fiber is called a *chromatid*. Depending on the phase of the cell cycle, a nucleus contains one or two distinct copies of a chromatid, building a *chromosome*.

The amount of chromosomes and potential abnormalities, both together called *karyotype*, describe the genetic characteristics of an organism. Hereby, different organisms or cell types have a varying number of distinct chromosomes as well as an individual amount of complete sets of chromosomes (*ploidy*). For instance, humans as diploid organisms typically have two complete sets of 22 sex unrelated chromosomes (autosomes) and two sex chromosomes (gonosomes). Deviations from this karyotype emerging at the replication of gametes or fertilization have shown to cause severe physiological conditions [Down, 1995; Antonarakis *et al.*, 2004; Turner, 1938; Saenger, 1996]. On a more local cell-specific scale the genetic origin of some cancer types have been linked with variations in the karyotype [Thompson and Compton, 2011; Nicholson and Cimini, 2013; Vasudevan *et al.*, 2020].

The entire genetic material of an organism is called its *genome*. For completeness it should be noted that the genome consists of more DNA outside the nucleus (*extrachromosomal DNA*) [Gaubatz, 1990; Kim *et al.*, 2020]. Extrachromosomal DNA can be found in the cell plasma [Sin *et al.*, 2020; Ma *et al.*, 2021] even if the number of nucleotides is much smaller compared to the nucleus. Aside from DNA in different cell compartments, some genomes are even composed of different nucleotides than DNA. For instance, some viruses store their genetic material using *RNA*, a different set of nucleotides than DNA. Nonetheless, the genetic material of a virus is called a virus genome.

In brief, a genome and its set of nucleotides are the blueprint of the cell and many of its metabolites. The DNA of chromosomes is classified into *genes* and *intergenic regions*. As opposed to intergenic regions the genes yield the DNA sequence that is directly processed by the cell. *Processing* genes can involve multiple steps like *transcribing* DNA into RNA, deleting and rearranging RNA sequence fragments (*splicing*) or *translating* RNA into amino acids. Ultimately, all the intermediate and final products of DNA processing define the cell type, the cell's morphology and functionality.

DNA sequencing deciphers the code of life

Having the knowledge about DNA and genomes from the perspective of experimental biology one immediate challenge is to precisely characterise the nucleotide sequence of a particular organism. With the advancement of computers, micro technology and biochemical engineering a methodology called *DNA sequencing* evolved. DNA sequencing is the procedure of capturing the precise order of nucleotides in a DNA sequence and can be divided in two major steps. The first step is an *in vitro* preparation of the sample (e.g. blood, saliva, epithelial cells or tumor tissue). The second step is typically a combination of a chemical, mechanical and computational process that actually captures the nucleotide sequence.

The first widely applied sequencing technology was developed by Frederick Sanger in 1977. In the in *in vitro* phase of the *Sanger sequencing* protocol [Sanger *et al.*, 1977] the DNA was fragmented into 100-500 bp segments and gradually hybridized with fluorescent

dideoxynucleotides. In the second phase a chromatograph and a laser detector produced chromatograms whose peak sequence could be used to read out the nucleotide sequence.

However, the Sanger sequencing protocol had the drawback of being labor-intensive especially during the sample preparation. The yield of DNA sequence was barely suitable for the purpose of large scale sequencing like the sequencing of large genomes. This limitation motivated the development of further sequencing methods for higher throughput [Voelkerding *et al.*, 2009], commonly termed *next generation sequencing* methods (*NGS*). Today, the most prevalent and widely used NGS technology is commercialized by Illumina Inc., therefore sometimes called *Illumina sequencing*. Their core innovation is a combination of two biochemical reactions, *bridge amplification* on an oligonucleotide matrix and *sequencing by synthesis*, that enable signal amplification and a rapid read out of optical reaction signals, respectively. Within a machine (*sequencer*) the optical signals are captured and translated into their respective nucleotide sequence. NGS sequencing has the advantage of achieving a significantly higher throughput than Sanger sequencing [Pareek *et al.*, 2011]. With a modern sequencer, e.g. Illumina NovaSeq 6000, at maximum performance one can sequence about 48 entire human genomes in about 44 hours (see www.illumina.com/systems/sequencing-platforms/novaseq.html, accessed on April 28th, 2021). In addition, sequences generated with Illumina sequencers today are less prone to experimental and technical errors compared to the complex procedure of Sanger sequencing.

Nevertheless, the few NGS technologies that dominate the global market today have one striking disadvantage for the downstream data analysis. The up to billions of DNA sequences they generate are only very short pieces (called *reads*) of the given DNA sample. For instance, one single read of an Illumina sequencer is typically 150 bp long. On a bigger picture, this means that sequencers today do not explicitly generate the sequence of an entire genome or even chromosome but only tiny, local and randomly ordered snapshots of it. Over the course of decades many algorithmic approaches have been developed with the objective to piece together this enormous puzzle of reads into entire chromosomes (Figure 2). A DNA sequence that got reconstructed from multiple reads is called an *assembly* (chapter 2.5) and the applied algorithmic approach an *assembly algorithm*. Modern assembly algorithms [Simpson *et al.*, 2009; Zerbino and Birney, 2008; Bankevich *et al.*, 2012; Chikhi and Rizk, 2013] are highly complex, utilize many advanced algorithms and data structures [Idury and Waterman, 1995; Schleimer *et al.*, 2003] and require a large amount of compute resources. DNA assembly is formally defined in chapter 2.5.

Over the last decade the latest developments in sequencing technologies yielded novel *third-generation sequencing* technologies (e.g. Oxford Nanopore Technologies, Pacific Biosciences). Third-generation sequencing technologies and their mechanics differ a lot from the next generation sequencing methods. In brief, both leading technologies utilize pores, biological or artificial, to drag DNA sequences through and capture an electric current or light emission. The reads that are generated form third-generation sequencing technologies (often termed *long-reads*) are substantially longer than the reads from NGS

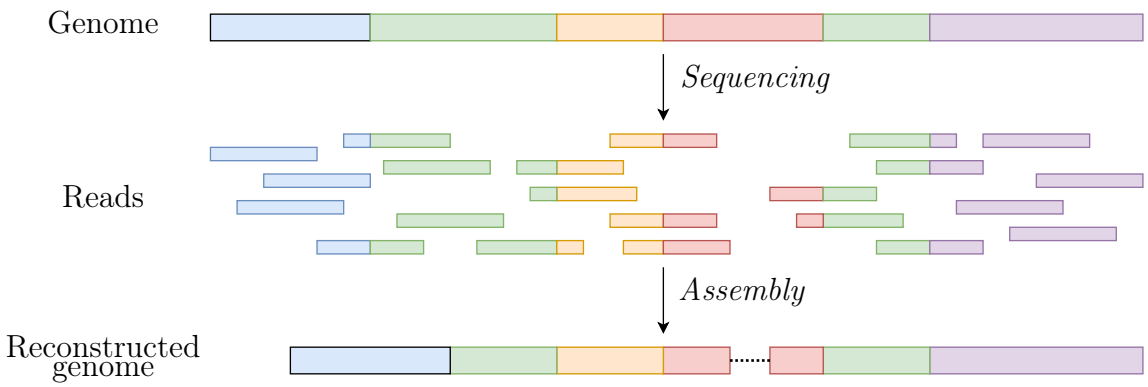

Figure 2: Sketch of sequencing and reconstructing a genome. The precise base pair sequence of the genome (on top) is unknown. When a genome is *sequenced*, e.g. with the NGS sequencing technology, it results in many reads covering fractions of the genome (middle). The reads are then used to reconstruct the genome, called *genome assembly*. The final assembly of a genome (bottom) might differ from the actual biological sequence of the genome. Some fractions of the assembled genome might differ in their length compared to actual sequence (left green block). This typically happens when the same sequence occurs multiple times in the genome (green blocks) or the sequence itself has a highly repetitive base pair pattern. Other fractions of the sequenced genome are potentially missing in the reconstructed genome (dotted line) if no reads captured these base pairs during the sequencing.

technologies. Long-reads typically show an average read length of 10-30 kbp [Amarasinghe *et al.*, 2020] but can reach up to over two million base pairs [Payne *et al.*, 2019]. Longer reads tend to facilitate the assembly of even longer sequences making long-reads the best data resource to assemble whole chromosomes [Phillippy, 2017]. However, state-of-the-art third-generation sequencing technologies have a comparably high *error rate*, i.e. the individual bases in the read are more likely to be incorrectly interpreted from the raw signal of the sequencer.

The exploration of the human genome and its diversity

With the technology of DNA sequencing in hand the National Institute of Health (NIH) sought to sequence the entire human genome for the first time in history. From 1988 on the NIH started to gather scientists for the *Human Genome Project* and a year later founded the National Center for Human Genome Research (which later became the National Human Genome Research Institute, short *NHGRI*). Over the course of eleven years (1990-2001) many researchers at the NHGRI sequenced and assembled the human genome. About the same time an American company named *Celera Genomics* led a privately funded effort to outpace the Human Genome Project for the first draft of the

human genome. Ultimately, in 2001 both sources published [Lander *et al.*, 2001; Venter *et al.*, 2001] their early genome drafts. These early genome drafts, together with multiple succeeding sequencing efforts over the years, have evolved and have been refined into what is known as the *human reference genome* today.

The breakthrough of high-throughput sequencing using NGS has substantially facilitated the sequencing of individual genomes since the era of the Human Genome Project. Today individual human genomes can be sequenced within a day with modern sequencers. Therefore, recent projects [Sherman *et al.*, 2019; Turnbull *et al.*, 2018; Byrska-Bishop *et al.*, 2021] comprise many hundreds or thousands of genomes of moderate quality. For instance, the *Thousand Genomes Project* (1KGP) in its latest phase [Auton *et al.*, 2015; Sudmant *et al.*, 2015; Byrska-Bishop *et al.*, 2021] comprises over 3,200 human genomes. As the human reference genome is mostly assembled from the DNA sample of a single anonymous African-American donor [Goldfeder *et al.*, 2017] it lacks ethnic and consequently genetic diversity. In contrast, the 1KGP contains samples from five different continental super-populations. Such a diverse repertoire provides a valuable resource to investigate the incidence of genetic traits between and within populations. In 2018 an even larger project [Turnbull *et al.*, 2018] was completed by sequencing 100.000 genomes in the United Kingdom. The scale of the *100.000 genomes project* enables advanced diagnosis and personalized treatments of rare diseases and cancer.

A latest trend in genomics is the large-scale sequencing of individual demographic groups [Liu *et al.*, 2015; Mallick *et al.*, 2016; Hehir-Kwa *et al.*, 2016; Maretty *et al.*, 2017; Kehr *et al.*, 2017; Wong *et al.*, 2018; Sherman *et al.*, 2019; Eisfeldt *et al.*, 2020; Taliun *et al.*, 2021], in case of country-specific studies sometimes called *national cohorts*. These sequencing projects often investigate how certain demographic groups differ from the reference genome, whether they have characteristic, overabundant genes and how this affects susceptibility or resistance to certain diseases.

The long and short of DNA variants

Single nucleotide variants. As opposed to deviations in the karyotype, i.e. gain or loss of an entire chromosome, it is much more common that the DNA sequence itself differs from the reference genome and between individuals. If such a difference comprises the substitution of only one singe nucleotide it is termed *single nucleotide variant*, short *SNV*. A substitution of one singe nucleotide that specifically affects in the germline is termed *single nucleotide polymorphism*, short *SNP*. More general, nucleotides can not only be substituted but also be inserted or deleted at a certain position in the strand. This insertion or deletion of a single or few nucleotides is commonly abbreviated *indel*. Due to a plethora of mechanisms involved that process DNA to their intermediate or final products in a cell [Crick, 1970; Fredrick and Ibba, 2009; Rédei, 2008; Chang *et al.*,

2007] SNVs are not necessarily harmful. For instance, some SNVs cause *silent mutations* [Zheng *et al.*, 2014] where the variants can go unnoticed by the cell.

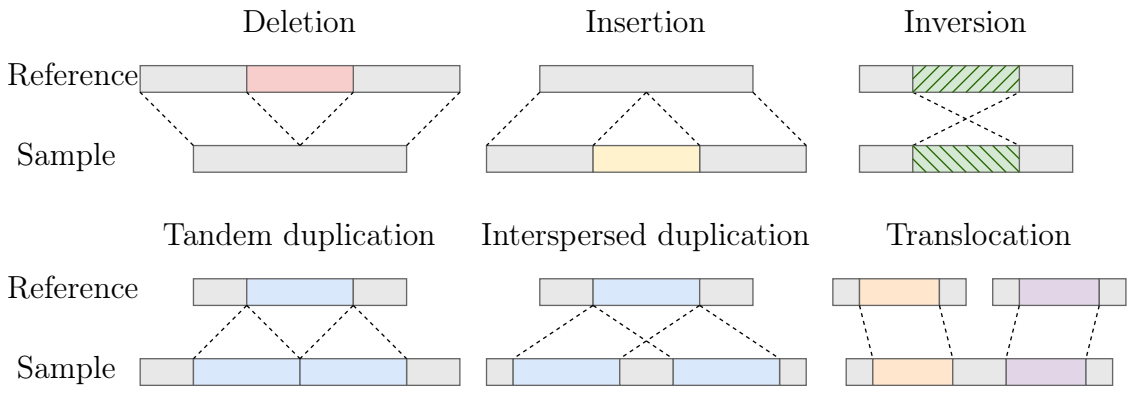

Figure 3: Types of structural variants between a reference and a sample. Types of structural variants are deletions (red), insertions (yellow), inversion (green), duplication (blue) and translocation (orange and purple). As opposed to SNVs, structural variants span a larger number of base pairs.

Structural variants. In addition to variants that affect only a single nucleotide, a DNA strand can also be subject to variants that affect whole segments, i.e. larger consecutive chains of nucleotides, called *structural variants* (SV). In the literature [Alkan *et al.*, 2011; Ebert *et al.*, 2021], the threshold of 50 bp is commonly used to distinguish between smaller indels and SVs. Today, many studies [Ho *et al.*, 2020; Mahmoud *et al.*, 2019; Willson, 2020] discovered and described various classes of SVs (Figure 3): sequence that is present in the reference genome but missing in a sample (*deletion*), sequence that is present in the sample but missing and unknown in the reference genome (*insertion*), sequence that has a reverted orientation in the sample compared to the reference genome (*inversion*), sequence of the reference genome that is copied multiple times consecutively to one position or non-consecutively to many positions of the sample (*tandem duplication* or *interspersed duplication*, respectively) and sequence that is present in the reference genome but appears at a different position, possibly different chromosome, in the sample (*translocation*). Since SVs affect many more base pairs than SNV they occur less frequently but often times have a more severe effect [Wang *et al.*, 2020; Collins *et al.*, 2020] especially when present in gene coding regions of the DNA [Walsh *et al.*, 2008; Collins *et al.*, 2020]. Multiple studies [Carvalho and Lupski, 2016; Mahmoud *et al.*, 2019] observed that SVs can occur combined at the same time. For instance, some inversions were found to be flanked by deletions or tandem duplications [Mahmoud *et al.*, 2019]. Also, recent findings describe more complex chromosomal rearrangements [Meyerson and Pellman, 2011; Fukami *et al.*, 2017] that can be dissected into multiple canonical SVs.

Non-reference sequences. Figure 3 shows that insertions are the only type of variant where a sample contains DNA sequence that is absent from the reference genome (*non-reference sequence, NRS*). Such NRS are found in every individual [Faber-Hammond and Brown, 2016] and are occasionally associated with a disruption of DNA processing of the cell [Wong *et al.*, 2020]. Insertions are a less frequently investigated class of variants as their detection inevitably requires an assembly (follows in chapter 2.5) of the NRS. Aside from the computational complexity of sequence assembly a majority of NRS contains highly repetitive DNA sequence [Delage *et al.*, 2020; Manni and Zdobnov, 2020] which additionally complicates assembly.

Some references [Delage *et al.*, 2020; Wong *et al.*, 2020] refer to the insertions of non-reference sequences as *novel sequence insertions* because the insertions describe a novel sequence content with respect to the reference genome. However, it has been found [Lee *et al.*, 2020; Kehr *et al.*, 2017] that a vast majority of NRS in humans is ancestral sequence from other primate genomes rather than actual novel sequence. A convincing theory is that NRS are not inserted into particular genomes but rather that the reference genome lacks diversity and therefore misses certain human sequences (called *reference bias*).

Alleles and genotypes. The characteristic form of a variable genomic locus, which might be equal to a given reference or affected by one or more variants, is called an *allele*. Related to the ploidy of an organism or cell type a gene can coexist as different specific alleles. For instance, a healthy human tissue cell has a diploid chromosome set and therefore contains two alleles of a distinct gene. Thus, with respect to the reference genome there are three different possible combinations of the two alleles (*genotypes*): both alleles are identical to the reference and do not carry a variant (*homozygous* reference), one allele is identical to the reference while the other allele differs from the reference by one or more variants (*heterozygous*) or both alleles differ from the reference (*homozygous* alternative).

Importance of variants. The immediate questions once variants have been identified is how they affect disease and diversity. DNA variants play an important role for the susceptibility and resistance to certain diseases [Leffler *et al.*, 2017; Dolatabadian *et al.*, 2020] and phenotypes [White and Rabago-Smith, 2011]. For instance, the *MHC complex*, which is one of the key components of the human immune system, is highly dependent on its genetic diversity to maintain its functionality against a variety of pathogens and viral agents [Norman *et al.*, 2017]. Other variants are associated with a reduced or increased risk to develop medical conditions [Pritchard *et al.*, 2016]. Variants can also be associated with morphological traits like eye color [White and Rabago-Smith, 2011] or limb malformation [Spielmann *et al.*, 2018].

Spread of variants. A variants that is present in a gamete is called a *germline variant* and can be passed on to an offspring generation. If the variant is passed to the offspring that variant gets incorporated into the DNA of every cell of that individual. In contrast,

somatic variants initially occur in a single somatic body cell and proliferates to identical *mutant cells*. A growing population of viable but severely dysfunctional mutant cells is what is termed *cancer* [Li *et al.*, 2020].

DNA variants aid the characterization of genetic diversity

The previous section introduced single nucleotide variants, indels and large structural variants. As shown in Figure 3 the position, orientation and length of a variant is the result of a sequence comparison between two DNA sequences. A fundamental method to determine sequence similarly and variants is called *sequence alignment* (explained in more detail in chapter 2.2). Sequence alignments [Smith and Waterman, 1981; Needleman and Wunsch, 1970] optimize a weighted scoring function to report on the similarity of two or more strings, e.g. DNA sequences. In the case of DNA sequences the individual matching bases and dissimilarities between the sequences determine a total *alignment score*. The alignment score of a DNA sequence alignment is a quantification of the similarity between the sequences. The *alignment* itself, the layout of pairwise matching bases, characterizes matching and mismatching fractions of the sequences.

An example for the wide spectrum of applications that utilizes sequence alignments is to determine the relationship between species. The general idea is that close related species share a larger amount of common DNA sequences whereas variants, in this context also called *mutations*, indicate evolutionary separation. A very prominent model that describes evolutionary relationships is called a *phylogenetic tree*. Phylogenetic trees cluster individuals or species into groups of common ancestors based on their sequence conformity and mutations. Close related but also some seemingly different species share a vast amount of sequence [van Dijk *et al.*, 2001; Yazhini *et al.*, 2021]. Shared sequence among species derived from a common ancestor it is called *homologous*.

The era of DNA sequencing shapes medical diagnostics and treatments

The enormous amount of sequence data being available today provides insight into the genetic origin of many diseases [Kingsmore and Saunders, 2011; Posey, 2019; Liu *et al.*, 2019] and subsequently a plethora of variants with medical implications [Stankiewicz and Lupski, 2010; McColgan and Tabrizi, 2018; Schüle *et al.*, 2017; Wilfert *et al.*, 2021] has been found over the last decades. In recent years it has become a prevailing method to sequence individuals with phenotypes of a certain disease or trait and compare the samples to individuals without the phenotype. Ideally, if this comparison yields a significant correlation between a variant and a phenotype the variant is called *associated* with the disease or trait. When initially the variants of the entire genome are taken into consideration, these comparative studies are termed *genome-wide association studies* (GWAS). For instance, a GWAS [Meddens *et al.*, 2020] of the relative intake of macro nutrients has discovered fourteen SNPs that correlate with phenotypes of obesity, type 2

diabetes and heart disease. Another GWAS [Weuve *et al.*, 2018] has associated non-carriers of the *APOE ε4* allele with an increased risk for Alzheimer's disease and dementia.

The catalogues of variants identified with GWAS have shown to be a valuable and sustainable resource beyond their initial generation [Kehr *et al.*, 2017; Jakubosky *et al.*, 2020]. A study of the Icelandic population identified a set of structural variants in non-repetitive regions of the human genome that is absent from the human reference genome [Kehr *et al.*, 2017]. A few of the detected SVs strongly correlate with known variants from the EMBL-EBI GWAS catalogue [Buniello *et al.*, 2019] and hence indicate associations with disease phenotypes, e.g. with age-related macular degeneration [Yang *et al.*, 2006] and Psoriasis [de Cid *et al.*, 2009].

Today, DNA sequencing has become an important factor for modern medicine and emerging medical treatments. A recent success in the treatment of *sickle-cell anaemia* [Ledford, 2020] has been driven by the molecular insights into human DNA. Promising results were shown from a bone-marrow transplantation of a *CRISPR-Cas9* gene-edited stem and progenitor cells [Frangoul *et al.*, 2020]. Another method [Esrick *et al.*, 2020] for the same disease directly targets RNA. Both approaches target the cell pathway of the *BCL11A* transcription factor and upregulate the γ-globin production. Ultimately, this stabilizes a healthier production level of fetal hemoglobin.

Another recent and salient benefit of DNA sequencing has been surveillance in a pandemic outbreak [Oude Munnink *et al.*, 2021]. The rapid spread of the *COVID-19 coronavirus* has globally challenged healthcare systems and demanded an immediate intervention. Here, DNA sequencing and subsequent variant detection has helped twofold. First, it was used to decipher the virus' RNA sequence [Zhou *et al.*, 2020; Zhu *et al.*, 2020; Sarma *et al.*, 2021] which is essential to start the pharmaceutical research on vaccines. With the RNA sequence in hand conventional clonal vector vaccines (AstraZeneca PLC, Johnson & Johnson Inc.) and novel mRNA vaccines (BioNTech SE, Moderna Inc.) could be developed and manufactured. Second, frequent or routine sequencing of infected patients has given insight into mutations of the virus [Harvey *et al.*, 2021]. The global surveillance of virus mutations is important to estimate the contagiousness of emerging variants and whether they might escape the immunization of certain vaccines.

Outline of the thesis

The previous explanations have illustrated that DNA sequencing and the subsequent genomic sequence analysis is an essential building block in many fields of biology and modern medicine. However, as part of the genomic sequence analysis, the detection of SV in the genomes of newly sequenced individuals remains challenging, particularly with the short reads of NGS technologies.

From data analyses [Alkan *et al.*, 2011] and methods developed in recent years [Kehr *et al.*,

2016; Kavak *et al.*, 2017] it was found that the detection and characterization of NRS can benefit from the processing of genomic sequence data from multiple individuals at once. Still, NRS have received less attention compared to other types of SV and there is only a small number of applications which can efficiently process the vast amount of NGS data from multiple individuals. These observations (section 4.1.1) revealed a demand for methods to efficiently detect NRS from large groups of individuals and motivated the contributions to the field presented in this thesis.

Thesis structure. In brief, the methods chapter describes how a highly efficient data structure [Holley and Melsted, 2020] can be utilized to substantially improve the computational performance of a previous application [Kehr *et al.*, 2016] while maintaining comparable results and improving in certain aspects. A practical implementation in C++ is provided for the described methods that scales to, till this day, unmatched input data sizes. A preceding chapter provides definitions, terminology and theoretical preliminaries. The last chapter summarizes and discusses the thesis, draws conclusions about the methods and results and provides an outlook on potential future research in the field.

In detail, chapter 2 provides descriptions and definitions of key concepts in set theory, genetics and genomic sequence analysis that are essential for the understanding of the remaining chapter. Moreover, the de Bruijn graph and its derivations are explained in greater detail. The final two subchapters describe previous applications which were fundamental elements for the development of the presented methods.

Chapter 3 provides a literature review of SV detection methods and highlights projects which applied these methods to large groups of individuals. At first, a broad spectrum of classification criteria is introduced to distinguish SV methods and to determine their precise field of application. Methods for the detection of NRS are assessed in more detail with a focus on their application to many individuals. The last subchapter highlights the extend and findings of projects which applied NRS detection to many individuals.

Chapter 4 presents new contributions to the field of NRS detection. The first subchapter introduces the observations which motivated the development of a new method and a description of its general objective. Next, the approach and objective of the new method is presented by a precise problem formulation together with a greedy heuristic to generate a solution, i.e. a set of NRS. Finally, a software (*PopIns2*) that implements the many individual steps of the new NRS detection method as well as its parameters and optional features are presented in the last subchapter.

Chapter 5 showcases and examines the application of PopIns2 to simulated and real NGS data. First, the accuracy of the NRS detection is evaluated using NRS data of simulated, individual human genomes. A use case with real human NGS data is presented and evaluated using the Polaris Diversity Cohort and Polaris Kids Cohort. The application of

PopIns2 to 1000 Icelandic genomes shows its unprecedented scalability for NRS detection using data of large study groups and populations.

The final chapter 6 summarizes the contributions of the thesis and classifies them in the context of ongoing research in the field. The concluding comments reflect on the findings and achievements presented in the previous chapters and discuss the advantages and limitations of the presented methods. An outlook hints potential directions for future research on the topic.

2 Definitions and preliminaries

This second chapter introduces notations and concepts that are essential for the subsequent methods section of this thesis. First, common notations for sets and its operations are introduced. The second and third subsections provide formal descriptions for foundations in biology and bioinformatics like DNA sequences, sequence alignment and inheritance. In the forth and fifth subsections, particular algorithms and data structures used in the methods section are explained. Finally, the last two subsections describe two programs and their implementations as they are used in the methods.

The first three subsections introduce very basic concepts in computer science and biology that a reader from these fields might wants to skim. The chapters 2.4 and 2.5 explain the concepts of a *de Bruijn graph* and sequence assembly in more detail which are rather known to an experienced audience. The last two subsections yield the foundation to a deep understanding of chapter 4 and are highly recommended for any reader.

2.1 Sets and Vectors

Sets. An unordered *set* of unique elements is denoted with curly braces. The *cardinality* of a set X is denoted as $|X|$. The formulation $X = \varnothing$ denote an *empty* set. An element x of set X is denoted $x \in X$. Is an element x not present in set X it is denoted $x \notin X$. A set Y is a subset of X if and only if every element $y \in Y$ is also an element of X. $Y \subseteq X$ denotes Y being a subset of X.

Further, common set operators apply:

- The *intersection* of two sets X and Y is

$$X \cap Y : \{x : x \in X \text{ and } x \in Y\}$$

- The *union* of two sets X and y is

$$X \cup Y : \{x : x \in X \text{ or } x \in Y\}$$

Some evaluations in chapter 5.1 assume the validity of common set properties and operations (commutative law, set difference, etc) which can be found in [Cormen *et al.*, 2009] at appendix B.1. Two sets are called *disjoint* if $X \cap Y = \varnothing$. The *Jaccard Index* [Jaccard, 1912] of two sets X and Y is a measure of similarity of the sets and is defined as the ratio of their set intersection divided by their set union.

Vectors. Within the scope of this thesis the term *vector* refers to its definition in computer science, i.e. here a vector is a container for an ordered sequence of elements. A vector whose elements are of the basic arithmetic data type boolean is called *bitvector*. The Jaccard Index of two equally long bitvectors x and y is defined as

$$J(x, y) = \frac{\sum_i x_i \wedge y_i}{\sum_i x_i \vee y_i}$$

where x_i and y_i are the i-th elements in the vectors, respectively.

2.2 Genomic sequences, sequence alignment and insertions

Genomic sequences. A genomic sequence $s \in \Sigma^m$ is a character string (*string*) of length m over the alphabet $\Sigma = \{A, C, G, T\}$. The characters $\sigma \in \Sigma$ encode the four different nucleotides. The formulation $|s|$ of a string s denotes the length of s, i.e. its number of characters. The *reverse* of a string is the string in its inverted order. The *complement* of a genomic sequence is built by a transition function f_c that converts every character $\sigma \in \Sigma$ as follows: $f_c(A) = T$, $f_c(C) = G$, $f_c(G) = C$ and $f_c(T) = A$. The *reverse complement* $\bar{s}$ is the transformation of a string s after both operations, i.e. inverting the string and applying f_c to every character in s. A string in its given and reverse complement form are termed to be in forward (+) and backward (-) *orientation*, respectively. Further, the *canonical representation* $\tilde{s}$ of a string s is defined as the lexicographically smaller representation of s and $\bar{s}$. A *dimer* d is a short genomic sequence with $|d| = 2$.

A *prefix* of a string s is a substring that starts at the first position in s. A *suffix* of a string s is a substring that ends at the last position in s. Both a prefix and suffix is defined by its length l and therefore termed l-prefix and l-suffix, respectively. A genomic sequence of length k is called k-mer. Chromosomes are genomic sequences and a genome is a set of chromosomes.

A *minimizer* [Schleimer *et al.*, 2003; Roberts *et al.*, 2004] is the smallest substring of length m of a character sequence according to an order function f (Figure 4). Unless stated otherwise, a minimizer in this thesis will be the smallest substring according to their lexicographical order.

A *repetitive* sequence $s = p_1, p_2, ..., p_n$ is a genomic sequence that consist of multiple consecutive instances of a certain base pair pattern $p = b_1, b_2, ..., b_m$. In real biological data, s potentially contains instances $\tilde{p}$ of the pattern p that have a different base $\tilde{b}_k$ than

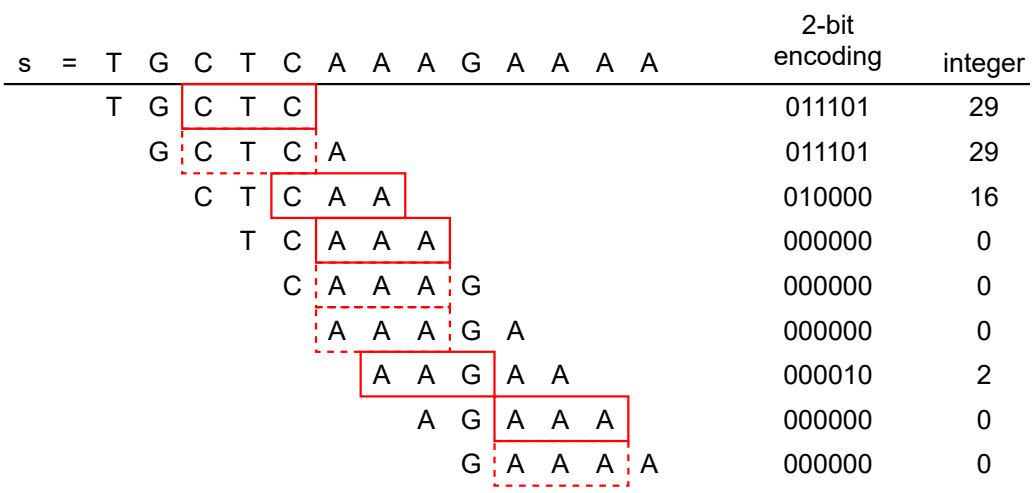

Figure 4: Sketch how to determine minimizers from k-mers of a genomic sequence s. First, a sliding window size k and a minimizer length m are chosen with the condition $m \leq k$. For every consecutive k-mer (here $k = 5$) in s every subsequence of length m (here $m = 3$) is translated into an integer via a 2-bit base pair encoding, i.e. $A \mapsto 00$, $C \mapsto 01$, $G \mapsto 10$ and $T \mapsto 11$. The subsequence of length m that translates to the smallest integer (given as 2-bit encoding in the middle column and as integer in the right column) is reported as a minimizer (red boxes) for that particular k-mer. For instance, for the first k-mer TGCTC the substring CTC translates to the smallest integer 29 among the other substrings TGC (57) and GCT (39) of length 3. Note that consecutive k-mers have $k - m$ overlapping subsequences of length m. Therefore, consecutive k-mers likely share their minimizer. The minimizers in the solid red boxes are the ones that are newly discovered with respect to the previous k-mer. The minimizers in the dashed red boxes are the ones that are equal to the previous k-mer.

b_k at the same pattern position k, where $1 \leq k \leq m$. A pattern instance $\tilde{p}$ is considered to be similar enough to p, and therefore to be subsequence of s if $\sum_{k=1}^{m} \tilde{b}_k \neq b_k \leq \varepsilon$ for a given error parameter $\varepsilon \in \mathbb{N}^+$.

The *entropy* $e \in \mathbb{R}^+$ of a genomic sequence s is defined as

$$\sum_{p \in P} -p \cdot \frac{log(p)}{log(2)} = \sum_{p \in P} -p \cdot log_2(p)$$

where P ($|P| = 4^2$) is the set of all relative frequencies of dimers in s. A sequence s is said to have *low complexity* if $e < \theta$ for a given threshold θ.

Reads. Chapter 1 introduced the principle that sequencers do not capture the entirety of a chromosome at once but only small fragments at a time. These fragments are called

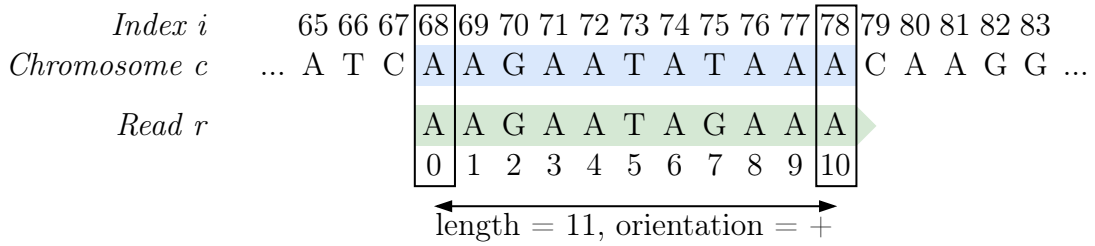

Figure 5: Sketch of a *read*. In bioinformatics a read r (green) is a subsequence of any given genomic sequence c (here a chromosome). These subsequences are generated by a sequencer. The origin (blue) of the read in c can be precisely described by its starting position in c ($i = 68$), its length (11) and orientation (+). Note that in this example r is an exact copy of its origin in c. In real data individual bases may vary due to an error probability during the sequencing.

reads. Therefore, reads are substrings of chromosomes (Figure 5). Depending on the sequencing platform and the laboratory protocol in use reads can have a different expected length. Unless stated otherwise, in this thesis reads will refer to NGS reads of expected 150 bp length.

Every sequencer of the different vendors reports *base quality scores* for every identified nucleotide in a read. This base quality score refers to the predicted accuracy of the nucleotide being correctly identified given the measured signal (light emission or electric current). The NGS reads that are referred to in the methods section typically come from Illumina Inc *HiSeq* or *HiSeqX* sequencers and have a *base quality score* Q that is logarithmically related to the error probability P of the identified nucleotide:

$$Q = -10\ log_{10}P$$

This base quality scoring is called *Phred Quality Score*[1] and is scaled into the interval $[0, 50]$ of $\mathbb{N}^+$ ($P = \frac{1}{100{,}000}$ is considered the lowest error rate by the machine).

In brief, during NGS chromosomes are sheared into smaller fragments which are sequenced from each end. As a result, reads come in a pair of two per fragment, called *read pair*. The reads in a read pair are a l-prefix and l-suffix of the fragment and are oriented in forward and backward orientation, respectively (Figure 6).

Due to the requirements of some data processing steps (explained in chapter 2.5), chromosomes are typically sequenced with a large redundancy. This redundancy is created by sequencing many technical replicates of the chromosome such that every base pair is

[1] https://www.illumina.com/documents/products/technotes/technote_Q-Scores.pdf

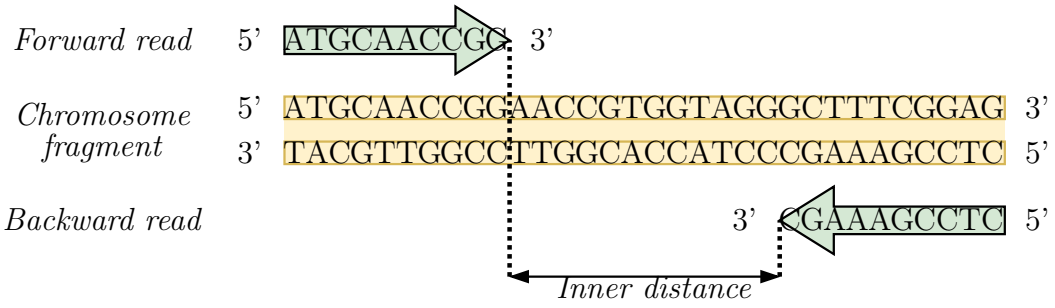

Figure 6: Schematic of a *read pair*. During the sequencing process long DNA strands are partitioned in smaller fragments (yellow). During NGS these fragments are sequenced from both ends resulting in a read pair (green). The internal mechanism of this sequencing technique always reads the strands of the fragment in $5' \to 3'$ direction. As a result, each read in a pair originates from one of the two complementary DNA strands. The *inner distance* is the number of base pairs in the fragment between the $3'$ ends of the reads in a pair.

captured by multiple reads (Figure 7). The number of reads that capture a particular base pair position is called *coverage*. The average coverage of the genome (*genome coverage*) C can be calculated with the *Lander-Waterman equation* [Lander and Waterman, 1988] as:

$$C = \frac{L \cdot N}{G}$$

where L is the read length, N is the number of reads and G is the length of the haploid genome.

Sequence alignment. An approximate match between two or more genomic sequences is commonly called *sequence alignment*. Sequence alignment is a fundamental and well explored field in bioinformatics. Even if used extensively, sequence alignment is not the main focus of the methods presented in this thesis and therefore only its fundamental idea is outlined here.

The essence of a sequence alignment is to decide whether two sequences are related to each other or similarity only occurred by chance. In order to make this decision the four key issues ([Durbin *et al.*, 1998], chapter 2.1) are which type of alignment to choose, how to score the alignment, the algorithm to find optimal or approximate alignments and a statistical method to evaluate the significance of the alignment score. The sequence alignments used in this thesis are mainly pairwise *local alignments* and *split alignments*.

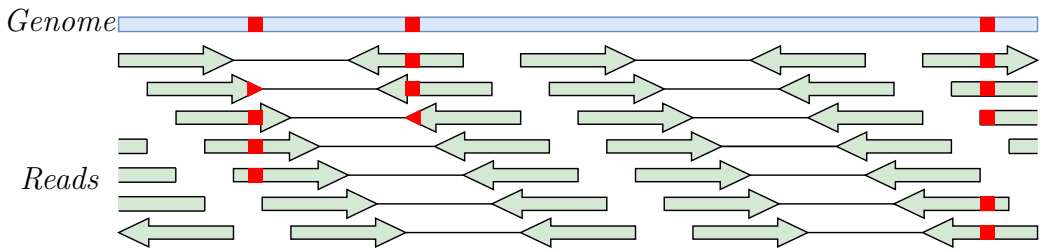

Figure 7: Read *coverage* of a genome. During NGS a genome (blue) or genomic sequence s is replicated and sequenced many times to achieve a random but approximately uniform distribution of read pairs (green) across s. The number of times a precise base pair position (red) in s is covered by a read sequence is called its *coverage*. The black lines connecting the reads of a pair denoted the inner distance.

A local sequence alignment of two genomic sequences x, y is to find the best alignment between subsequences of x and y ([Durbin *et al.*, 1998], chapter 2.3). In the workflows of chapter 2.7 and 4 the genomic sequences used for local sequence alignments are pairwise combinations of reads, longer contiguous genomic sequences and a reference genome. A well established algorithm for local sequence assembly is the *Smith-Waterman algorithm* [Smith and Waterman, 1981]. The original version of this algorithm uses *dynamic programming* [Bellman, 1957] where the final solution is incrementally computed via partial solutions. Given a fixed set of parameters, the Smith-Waterman algorithm is a deterministic method. In order to find one or multiple local sequence alignments between genomic sequences with a strongly uneven length, e.g. when aligning reads (short) to the human reference genome (very long), some methods [Li, 2013] implement a preceding indexing phase for the longer sequences. The indexing phase is utilized to rapidly identify potential alignment positions $\tilde{P}$ in the longer sequence before the positions $p \in \tilde{P}$ are verified with comparatively slower local sequence alignments.

Within the scope of this thesis the concept of a split alignment (Figure 8) and its implementation[2] is taken from the *seqAn* library [Reinert *et al.*, 2017]. A split alignment here is a sequence alignment between the genomic sequences x, y, y' where

1. y has an additional continuous subsequence s relative to x, such that the prefix to the left of s and the suffix to the right of s align to x and the bases of s in the center remain unaligned or

2. only a prefix of y and a suffix of y' align to the left and to the right of a particular position in x, respectively. The bases of y (y') not being part of the aligned prefix (suffix) remain unaligned in their entirety.

[2] https://docs.seqan.de/seqan/2.3.2/global_function_splitAlignment.html

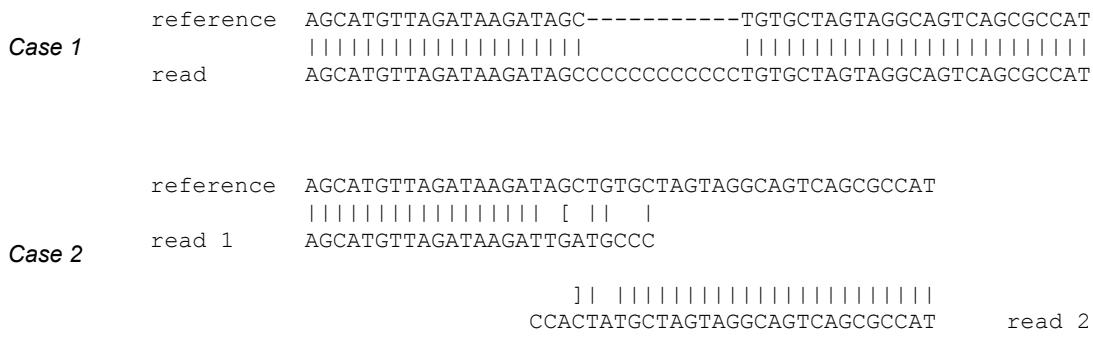

Figure 8: Split read alignments between reads and a reference. The "|" denote matches between the bases of a read and the reference. In Case 1 the read has an additional subsequence compared to the reference. The additional sequence in the read is unaligned while the flanking sequences to its left and right have alignments to the reference. In Case 2, read 1 has an alignment of its prefix until the position denoted with "[". The bases to the right of the prefix remain unaligned even though some bases match with the bases of the reference. Read 2 has an alignment of its suffix until the position denoted with "]".

Insertions. If not stated otherwise, *insertions* refer to genomic sequences of 50 or more base pairs that are present in a sequenced individual but absent at the corresponding location in the reference genome. An insertion is defined by an *insertion breakpoint* on the reference genome, the genomic sequence that is added at the insertion breakpoint and an *orientation* of the added genomic sequence (Figure 9). A set of insertion that was discovered from a single or multiple individuals is termed a *callset*.

The location of an insertion where genomic sequence is added with respect to a given reference sequence is called insertion breakpoint. An insertion breakpoint is defined by three identifiers: a reference the individual is compared to (typically a reference genome), a chromosome or sequence identifier if the reference comprises multiple sequences and either one or two base pair positions on the reference depending on whether the ends of the added sequence link to the same or different positions in the reference, respectively. The methods in chapter 4 refer to the *Human reference genome version 38* [3] which consists of a primary assembly of 22 autosomes as well as the X and Y sex chromosomes.

The type of genomic sequence that is present in an individual but missing at the corresponding location in the reference genome is generally not clearly defined. Some studies [Ebert *et al.*, 2021] count duplications and larger tandem repeats as insertions as these sequences are per se inserted with respect to the reference. Others studies [Kehr *et al.*,

[3]downloaded from ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000001405

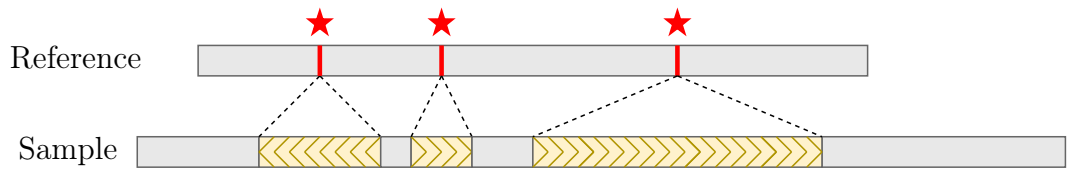

Figure 9: Insertions comprise additional genomic sequences (yellow) with respect to the reference genome (grey). The additional genomic sequence of an insertion is located at a precise base pair position (red star) in the reference genome, called the *insertion breakpoint*. The dark yellow arrows indicate the orientation of the additional genomic sequence.

2017] are more stringent and constrain the definition of the inserted genomic sequence to not be present elsewhere in the reference genome, therefore called *non-reference sequence* (NRS). Within the scope of this thesis the genomic sequences of insertions refer to NRS.

NRS can be present in an individual in two possible orientations, forward and reverse-complement. Since an insertion is defined as not being known in the reference the insertion's orientation of the NRS can indeed be determined but here the distinction of forward and reverse-complement relies on how the NRS is stored in the callset. NRS are commonly stored either in their canonical representation or in the orientation they were discovered first. Chapter 2.7 describes a method that determines the orientation for every NRS of a given callset that is present in a particular individual.

2.3 Genotype and inheritance

Genotype. The *genotype* of a genomic variant is the description of how many of the chromosome sets of the cell carry that particular variant. Each allele has the variant either present or absent which is commonly binary encoded with boolean values 1 or 0, respectively. A somatic non-cancer cell in the human body is typically diploid and therefore variants can occur in the genotypes homozygous reference (0/0), heterozygous (0/1) and homozygous alternative (1/1).

The process of determining the genotype of a variant is called *genotyping*. Genotyping methods commonly implement statistical models (e.g. as part of the method in chapter 2.7) that assign predicted likelihoods $g \in \mathbb{R}$ to each genotype, called *genotype likelihoods*.

Mendelian Inheritance. Later in this thesis the accuracy of the predicted genotypes for the NRS detected with the method introduced in chapter 4 will be evaluated using the first two *Mendelian inheritance* rules [Mendel, 1865] for single genetic traits:

1. *Law of dominance and uniformity.* If two organisms that differ from each other in just one characteristic are crossed, then the resulting hybrids F_1 are uniform in the chosen characteristic.

2. *Law of segregation.* An intermediate crossing of F_1 hybrids results not in a uniform segregation of the characteristic in the F_2 generation but in a ratio 1:2:1.

Figure 10 shows how the Mendelian inheritance rules can be used along with the genetic information of a mother-father-offspring trio (called *family trio*, short *trio*) to evaluate the accordance with the expected ratios. Both first and second Mendelian inheritance rule implies that each allele is inherited from the parent to the offspring generation with a 50% likelihood.

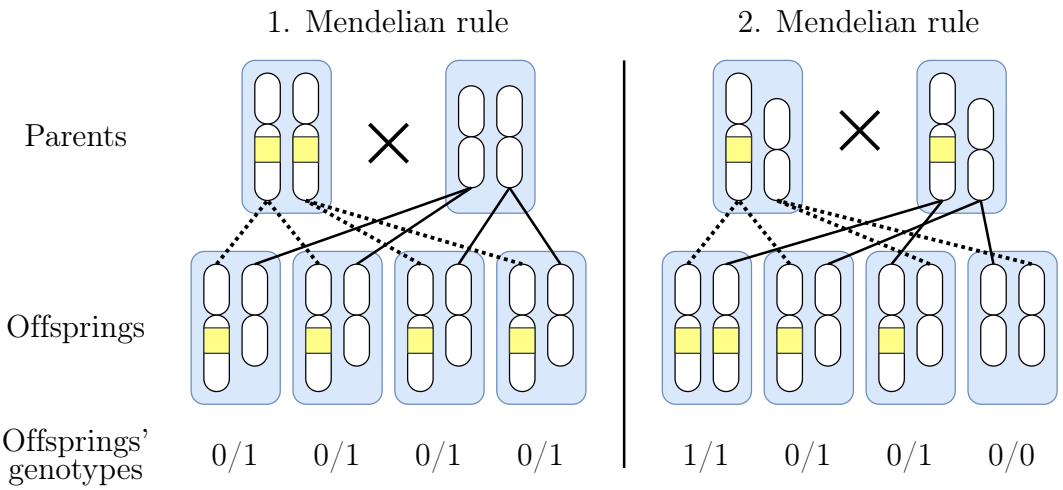

Figure 10: First and second Mendelian inheritance rule. The diploid chromosome set of a human cell (blue) consists of two chromatids (white) that separate during the gamete formation. Each chromatid potentially contains a structural variant (yellow). The first rule of Mendelian inheritance considers a parent generation of a homozygous carrier and a homozygous non-carrier (1/1 and 0/0). After crossing of the gametes (black lines) all offspring cells are heterozygous carriers (0/1). The second rule of Mendelian inheritance considers a parent generation of heterozygous carriers. After a crossing of the gametes the offsprings exhibit a ratio of genotypes of 1:2:1 for homozygous carrier, heterozygous carrier and homozygous non-carrier, respectively.

Hardy-Weinberg equilibrium. The *Hardy-Weinberg equilibrium* (HWE) [Hardy, 1908] is the population-genetic law that the genotypes and the frequencies of alleles in a population remain constant from generation to generation. The HWE assumes a number of requirements about the population to be valid like a minimum size to neglect random fluctuation, no constrains for the mate choice, no occurring mutations, equal

survival and reproduction rates and no migration into or out of the population. While these assumptions are virtually impossibly to establish in population studies the HWE remains a useful modelling for the dynamics of alleles in a population.

For a population with known genotypes of every individual the relative occurrence of an allele in the population (*allele frequency*, AF) can be determined. Let p, q be the AFs for a genetic characteristic (e.g. a structural variant) to be present or absent, respectively. In a population of diploid individuals the HWE is

$$p^2 + 2pq + q^2 = 1$$

where each term $p^2, 2pq, q^2$ is a *genotype frequency*, i.e. the proportion of each genotype in the population.

2.4 Graphs

Undirected graph. An *undirected graph* $G = (V, E)$ consists of a set of *vertices* V and a set of edges $E = V \times V$. An *edge* $e \in E$ in an undirected graph is an unordered pair (u, v) of vertices $u, v \in V$.

Directed graph. A *directed graph* $G = (V, E)$ consists of a set of *vertices* V and a set of edges $E = V \times V$. An *edge* $e \in E$ in a directed graph is an ordered pair (u, v) of vertices $u, v \in V$.

In an edge (u, v) we call u a *predecessor* of v and v a *successor* of u. The *in-degree* and *out-degree* of a vertex is the number of predecessors and successors, respectively.

A *walk* $w = v_1, v_2, ..., v_n$ through a graph is a sequence of vertices such that every vertex v_{i-1} is a predecessor of v_i for all $1 < i \leq n$. A *path* $p = v_1, v_2, ..., v_n$ is a walk without circuits, i.e. there exist only distinct vertices in p. A path p is called *non-branching* if v_1 has out-degree 1, all vertices $v_2, ..., v_{n-1}$ have in- and out-degree 1 and v_n has in-degree 1. A non-branching path $p = v_1, v_2, ..., v_n$ is *maximal* if the in-degree of v_1 and out-degree v_n are both unequal to 1.

A *Hamiltonian cycle* w_{Ham} is a walk through G with the property that all vertices $v \in V$ exist exactly once in w_{Ham} except for $v_1 = v_n$. An *Eulerian cycle* w_{Eul} is a walk through G with the properties that the set of edges $E_{Eul} = \{(v_{i-1}, v_i) | v_{i-1}, v_i \in w_{Eul}, 1 < i \leq n\} \cup \{(v_n, v_1) | v_n, v_1 \in w_{Eul}\}$ is equal to E. In other words, a walk w_{Eul} is an Eulerian cycle through G if and only if w_{Eul} contains every edge of G exactly once and w_{Eul} starts and ends in the same node.

An *induced subgraph* G_{sub} is a graph that consist of a subset of vertices $V_{sub} \in V$ and all edges that connect the nodes of V_{sub} in G.

A *connected component* in an undirected graph is an induced subgraph where any two vertices are connected to each other by a path.

Path cover. Let $V_p \subseteq V$ be the set of vertices of a path p. The set $P = \{p_1, p_2, ..., p_m\}$ of paths through a graph $G = (V, E)$ is called a *path cover* if and only if each vertex $v \in V$ is present in at least one $p \in P$, i.e. $V = \bigcup_{p \in P} V_p$.

De Bruijn Graph. A *de Bruijn Graph* (DBG) is a directed graph $G = (V, E)$ built from a set of sequences S and a given k-mer size k, such that every k-mer that is a substring of either s or $\bar{s} \in S$ corresponds to a vertex $v \in V$. An edge $e = (u, v) \in E$ between the vertices $u, v \in V$ exists if and only if the $(k-1)$-suffix of u is equal to the $(k-1)$-prefix of v. A walk or path p of n vertices in a DBG corresponds to a genomic sequence $\omega(p)$ of length $n + k - 1$. The genomic sequence corresponding to a maximal non-branching path in a DBG is called a *unitig*.

A *compacted* DBG $G' = (V', E')$ is a de Bruijn Graph where every maximal non-branching path is represented as a unitig $v' \in V'$. In a compacted DBG G' an edge $e' = (u', v')$ between the two unitigs u' and v' exists if and only if the $(k-1)$-suffix of u' is equal to the $(k-1)$-prefix of v'. Figure 11 shows on a small example how a DBG is transformed into a compacted DBG. A *maximal unitig path* [Krannich *et al.*, 2021] $p = \{v'_1, v'_2, ..., v'_m\}$ through a compacted DBG is a path where the in-degree of v'_1 and out-degree of v'_n are both equal to 0.

Modern practical implementations of the de Bruijn Graph are often designed as a *bidirected* de Bruijn Graph. Bidirected de Bruijn Graphs account for the strand information of a vertex' sequence. Each vertex represents a k-mer (or unitig) and its reverse complement. An edge between two vertices is, as the name indicates, bidirected as each direction connects either the forward or reverse complement sequences [Medvedev *et al.*, 2007]. A bidirected de Bruijn graph complies with the definition of an undirected graph. Therefore, connected components can be determined in a bidirectional DBG. If not stated otherwise, a de Bruijn graph is always considered bidirected for the remainder of this thesis.

The definition of the DBG used in this thesis is called *node-centric* [Chikhi and Rizk, 2013], i.e. k-mer sequences are associated with vertices while the edges arise implicitly as a result of vertex overlaps of length $k - 1$. Other publications [Muggli *et al.*, 2017] use an *edge-centric* definition where edges (u, v) are associated with $k + 1$-mers and the nodes u and v arise implicitly as their k-prefix and k-suffix, respectively. Ultimately, both definitions result in the same graph topology.

Colored de Bruijn Graph. The de Bruijn Graph is defined over an input set S of one or multiple sequences. In contrast, a *colored* de Bruijn Graph (CDBG) $G = (V, E, C)$ is built from a set of sequence sets $\mathcal{S} = \{S_1, S_2, ..., S_n\}$. A CDBG extends the DBG by adding a label to every k-mer in G indicating which sequence sets $S_i \in \mathcal{S}$ contain the k-mer.

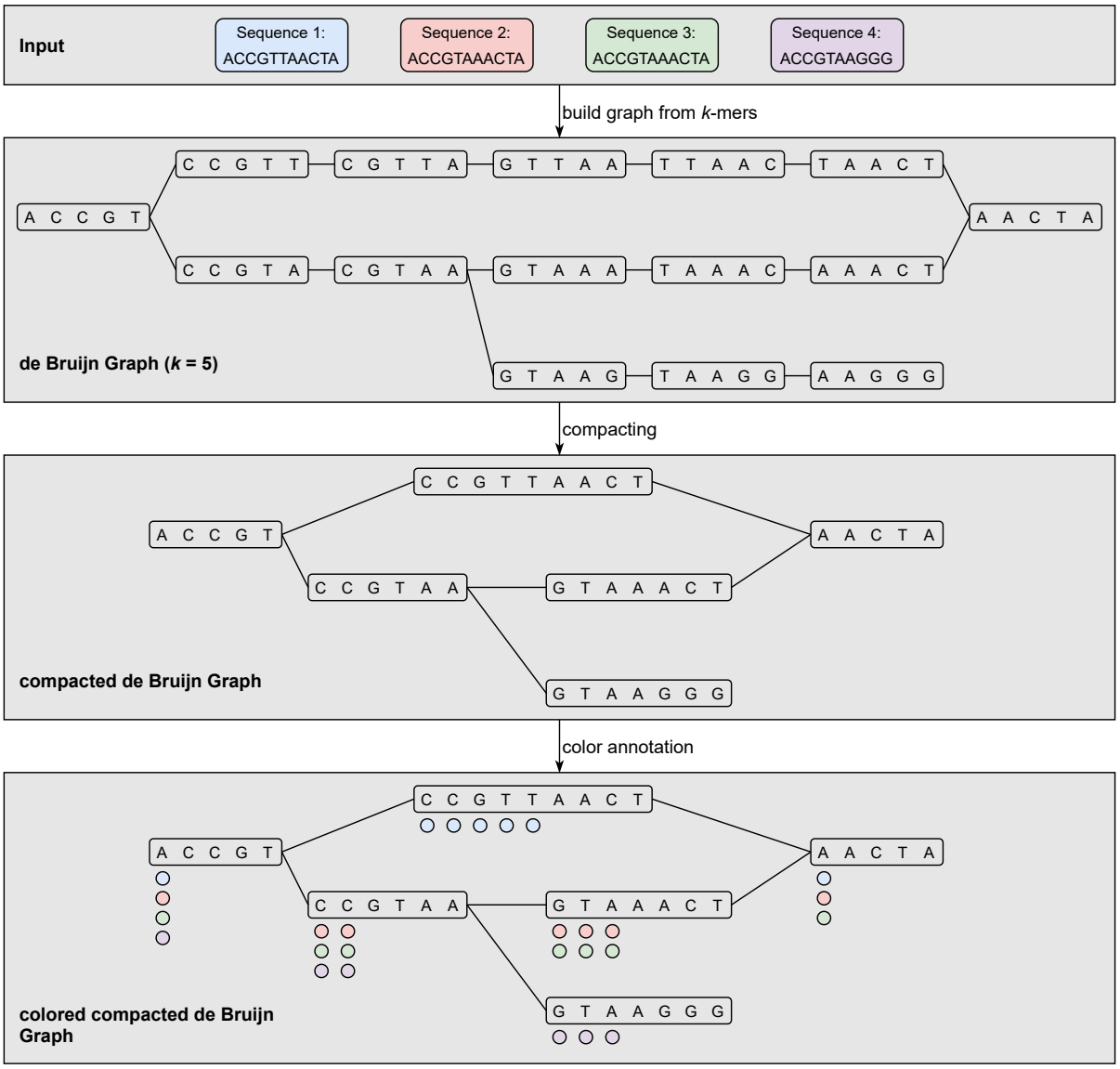

Figure 11: Sketch of de Bruijn Graph (DBG) types. The box at the top shows the input sequences S for the de Bruijn Graphs. The second box from the top shows a DBG with k-mer length five. Every continuous 5-mer of every $s \in S$ is a vertex in the DBG and edges exist if and only if two vertices have a continuous overlap of four bases. The second box from the bottom shows a compacted DBG where every maximal non-branching path p of vertices is compacted into one new vertex (unitig). The sequence of the unitig is $\omega(p)$. The box at the bottom shows a colored and compacted DBG (CDBG). Each k-mer in a CDBG is labelled with the corresponding input sequences $s \in S$ which contain the k-mer. In this example, the color labels indicating the input sequences are shown at the leading base of each k-mer.

The labels of the k-mers in a CDBG are commonly implemented with bitvectors. The i-th entry of the bitvector, with $1 \leq i \leq n$, denotes the presence (1) or absence (0) of the k-mer in sequence set S_i. A unitig of base pair length l contains $l - k + 1$ bitvectors, one for every k-mer. Therefore, a CDBG requires $(l-k+1) \times n$ additional bits of memory per unitig.

In practise, the bitvectors of all k-mers are often stored in a matrix $C^{n \times m}$, where n is the number of input sets and m is the number of distinct k-mers in $\mathcal{S}$. Every entry $c_{i,j} \in C$ indicates whether k-mer j is present in input set S_i. In the initial publication [Iqbal *et al.*, 2012] of the CDBG the bits were thought of as colors (Figure 11, lowest box). We therefore refer to the bitvectors as *color vectors* and the matrix C as *color matrix*.

2.5 Sequence assembly

The objective of a *sequence assembly* is to reconstruct a character sequence S from many of its much shorter subsequences. To obtain a complete S every character $c_i \in S$, with $1 \leq i \leq |S|$, has to be present in at least one of the subsequences. In order to piece together the subsequences correctly with respect to S without knowing S (called *de novo* assembly), the subsequences need to have sufficiently overlapping ends.

The assembly of genomic sequences underlays the same requirements. Here, S is usually a genome (called *genome assembly*) or, more general, one or multiple genomic sequences (called *DNA fragment assembly* [Pevzner *et al.*, 2001] or *local assembly*). The comparably short subsequences that are used to reconstruct one or many genomic sequences are the reads of a sequencer. The resulting contiguous sequences of an assembly are called *contigs*. To obtain a completely reconstructed (*assembled)* S every base pair $b_i \in S$, where $1 \leq i \leq |S|$, has to be present in at least one read. In chapter 2.2 it was noted that NGS typically produces enough reads for a multiple genome coverage C. The multiple read coverage of S generated by all modern high-throughput sequencing technologies, irrespective of whether it be NGS or third-generation sequencing, is vital for the assembly of any genomic sequence due to the following probabilistic properties:

- more reads (randomly sampled from S) will more likely contain every $b_i \in S$,
- a higher C will likely yield longer and more unique pairwise read overlaps and
- a poor base quality score, and potentially misinterpreted nucleotide, of a certain b_i in a read can be compensated and corrected with many confidently identified nucleotides in other reads that contain b_i.

The challenge in the assembly of genomic sequences is to piece together a set of reads R into one or multiple sequences $\tilde{S}$ that match the true S as much as possible. The majority of *de novo* assembly algorithms for this task can be categorized into the two following prevailing paradigms (Figure 12).

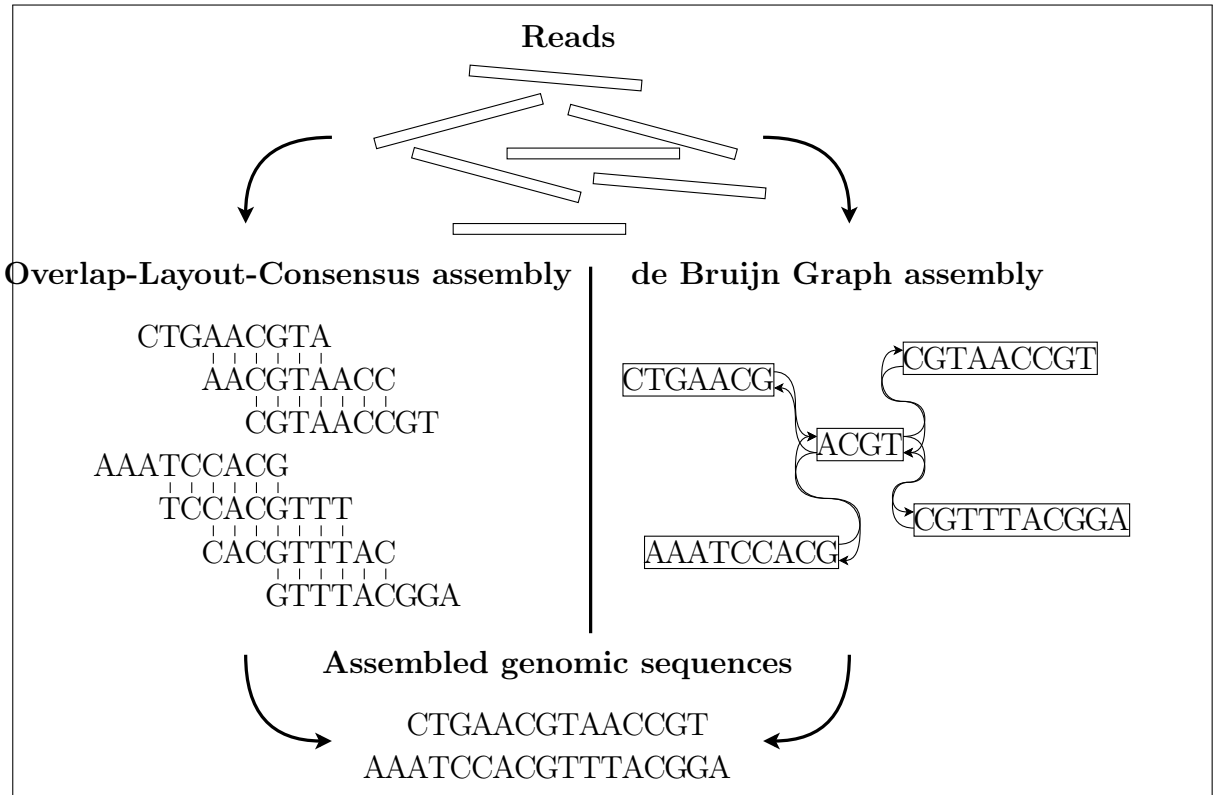

Figure 12: Assembly of genomic sequences. The two prevailing methods to assemble a set $\mathcal{R}$ of genomic reads (top) are the overlap-layout-consensus (left) and de Bruijn graph (right) paradigm. In brief, the overlap-layout-consensus methods are finding pairwise overlaps of the reads and gradually extent a group of matching sequences. Finally the consensus sequences (bottom) can be extracted. In contrast, the de Bruijn graph methods build a de Bruijn graph and extract consecutive unitigs. The concatenation of the unitig sequences are the assembled genomic sequences (bottom).

Overlap-Layout-Consensus. The *overlap-layout-consensus* (OLC) paradigm for the assembly of genomic sequences consists of three consecutive phases:

1. *Overlap.* In the overlap phase all pairs of reads $r_i, r_j \in R$ are aligned to detect overlaps of a given minimum length ℓ_{min}.

2. *Layout.* In the layout phase all r_i, r_j with an overlap $\ell_{r_i,r_j} \geq \ell_{min}$ are put together. Often times the result of the layout phase is an *Overlap graph* $G = (V, E)$ that stores reads in vertices and has edges (r_i, r_j) if the reads r_i, r_j satisfy $\ell_{r_i,r_j} \geq \ell_{min}$.

3. *Consensus.* In the consensus phase contigs are generated from the reads that were put together, e.g. by following walks and paths across G. The resulting sequence or set of sequences is $\tilde{S}$.

However, there are some substantial problems associated with the classical OLC paradigm and its corresponding assembly algorithms [Compeau *et al.*, 2011]. One problem is the number of reads in R. In the overlap phase every read is aligned to every other read which results in a quadratic increase of alignments. With modern sequencers easily generating billions of reads the native overlap phase would need to perform at least 10^{18} alignments. Another algorithmic problem (of the layout phase) is to find the sequence or set of sequences of S in G. For instance, early on the concept of a Hamiltonian cycle was used [Kececioglu and Myers, 1995; Adams *et al.*, 2000] for the assembly of comparatively small circular genomes [Fleischmann *et al.*, 1995]. But as the computation of a Hamiltonian cycle is NP-*complete* (proof in [Cormen *et al.*, 2009], chapter 34.5, "NP-complete problems") this approach does not scale to larger and more complex genomes, e.g. the human genome.

Assembly via de Bruijn Graphs. Another approach to DNA fragment or genome assembly is the utilization of a de Bruijn Graph. It starts with the construction of a DBG $G = (V, E)$ by extracting every consecutive k-mer of all reads in R. As the k long sequences of all $v \in V$ originated from reads $r \in R$ and all $r \in R$ are subsequences of S it means in reverse that the entire set V has to be used in order to reconstruct a complete S (given that R already covers S and that there are no $v \in V$ due to incorrectly identified bases in the reads $r \in R$ as they are uninformative to reconstruct S).

The usage of a de Bruijn Graph circumvents the need for a pairwise sequence alignment of all $r \in R$ as seen in the OLC paradigm. However, from a given DBG one still needs to find walks or paths that reconstruct S (equivalent problem of the layout phase).

An early approach [Pevzner *et al.*, 2001] for genome assembly used a DBG built from a set of reads in order to reconstruct S (their DBG was slightly modified by introducing multiplicity of edges). Their approach to generate $\tilde{S}$ is to find an *Eulerian cycle* in G for which there are known polynomial time algorithms [Hierholzer, 1873]. If the objective is to find multiple non-cyclic sequences ($|S| >> 1$) from a DBG other approaches to follow along the edges can be beneficial that use a *breadth-first search* [Chikhi and Rizk, 2013] or *depth-first search* [Krannich *et al.*, 2021].

Evaluation statistics. Once an assembly $\tilde{S}$ is generated, irrespective whether it was done with the OLC or DBG approach, an evaluation method is required to determine how accurate $\tilde{S}$ matches the true S. The *N50* assembly statistic [Lander *et al.*, 2001] is the length of the shortest contig (i.e. the contigs are sorted by length) at 50% of the total assembly length. The N50 is solely a measure for contiguity of the contigs and has no validity for the correctness of the assembly, i.e. whether the contigs are an accurate reconstruction of S. The *NG50* statistic [Earl *et al.*, 2011; Magoc and Salzberg, 2011; Mäkinen *et al.*, 2012] is very similar to the N50 but instead of using the assembly length the actual or approximated genome length is used. In order to compute the *NA50* assembly statistic [Gurevich *et al.*, 2013] all contigs are aligned to a given reference genome to identify erroneous contigs, i.e. contigs that have discontiguous alignments with

respect to the reference. Each junction of two discontiguous alignments is accounted for as a *misassembly*. Next, contigs with misassemblies are split, at the junctions and the individual subsequences (blocks) are realigned to the reference. After all junctions are split the set aligned contigs and blocks is used to compute the N50 statistic, where the assembly length also includes the unaligned contigs and blocks. The resulting value is the NA50 value. The *NGA50* assembly statistic [Gurevich *et al.*, 2013] is an adaption of the NG50 to the NA50 statistic. It uses the NG50 statistic instead of the N50 statistic in the final step of the NA50 computation.

2.6 Bifrost

Introduction. *Bifrost* [Holley and Melsted, 2020] is an efficient implementation of the colored and compacted de Bruijn Graph. The software can efficiently build, index and query either a DBG or a CDBG. The internal data structures (see paragraph below) and its software design were chosen to maximize the compute performance in terms of speed and memory consumption. Hereby, Bifrost can take advantage of multi-core processors as well as the processor instruction set (*SIMD* operations). Bifrost allows for a dynamic modification of its graphs, i.e. k-mers or entire sequences can be added to or removed from a graph without the need for a full reconstruction of the graph. DBGs in Bifrost are bidirected DBGs (chapter 2.4).

Application. Bifrost can be used as a binary (via *command line interface*) or as a C++11 software library (*application programming interface, API*). The following describes a selection of Bifrost's API limited to the functionality that is used in the methods section of this thesis. Note that the CDBG in Bifrost is a C++ template class that is derived from a DBG template class, therefore every function that exists for the DBG can also be applied to a CDBG but not vice versa.

Bifrost has a multi-threaded file I/O that can read one or multiple FASTA, FASTQ and GFA files. Optionally, the input can be effortlessly filtered to only include k-mers with a minimum abundance of ≥ 2. The Bifrost API has a *build* function that builds the compacted de Bruijn Graph from a preliminary data structure that stores the input k-mers. The building process also creates an index for the unitigs such that unitigs can be enumerated and when a k-mer query has a positive result (k-mer exists in DBG) the corresponding unitig is returned. Once a DBG is built, the compacted structure is always maintained if k-mers are added or removed from the graph. For a given unitig $u_{current}$ all predecessor and successor unitigs (collectively termed *neighbors*) can be obtained in constant time. For any given unitig the *strandness* can be retrieved, i.e. the current orientation of the unitig within a walk or path. By default, unitigs are stored in an arbitrary orientation. Only unitigs obtained as predecessors or successors have a predetermined strandness with respect to $u_{current}$. Bifrost additionally provides a small set of *simplification* operations on a DBG's topology. Functions are provided to remove

tips and *singletons* (Figure 13) from a DBG. For each k-mer in a CDBG the color vector can be retrieved and modified.

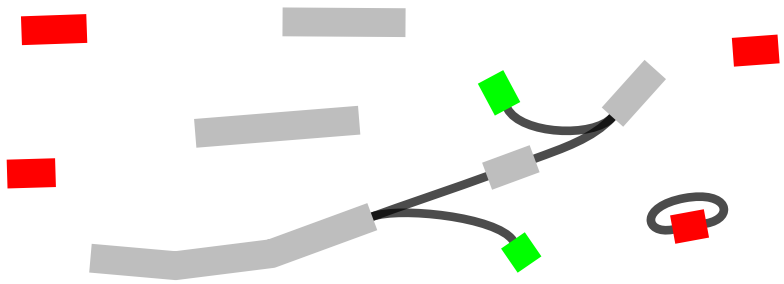

Figure 13: Compacted de Bruijn graph of several connected and unconnected unitigs (thick bars). Singletons (red) are unitigs u where $\{(v,u)|v \neq u\} \cup \{(u,v)|v \neq u\} = \varnothing$ and $|u|$ is smaller than a user defined threshold. Tips (green) are unitigs u where $|\{(v,u)|v \neq u\} \cup \{(u,v)|v \neq u\}| = 1$ and $|u|$ is smaller than a user defined threshold. The image was generated with *bandage* [Wick *et al.*, 2015].

Data structures. To build, index and query the CCDBG efficiently, Bifrost utilizes multiple carefully designed data structures.

In order to build a compacted DBG, Bifrost utilizes *Bloom filters* [Bloom, 1970] for fast membership queries of k-mers in the input data. A Bloom filter $\mathcal{B} = (B, H)$ consists of a bitvector B and a set of hash functions H, with $|H| \geq 1$. Initially all bits in B are zero. To insert a given set E of elements into $\mathcal{B}$, each hash function $h \in H$ generates a hash value between 0 and $|B| - 1$ for each element $e \in E$. Next, the bits in B at the positions of the hash values are set to one. A request whether an element $\tilde{e}$ is a member of $\mathcal{B}$, i.e. whether $\tilde{e}$ was previously inserted into $\mathcal{B}$, works in a similar way. All hash values for $\tilde{e}$ with the functions H are generated. If all bits in B at the corresponding positions of the hash values are ones then $\mathcal{B}$ returns true, i.e. $\tilde{e}$ is a member of $\mathcal{B}$. If any of the checked bits is zero then $\mathcal{B}$ returns false. Since the hash functions of a Bloom filter can potentially generate the same hash value for different elements (called *hash collision*), a positive result of a membership query has a chance to be a false positive. Bifrost implements additional strategies to speed up the operations of checking and setting the bits [Putze *et al.*, 2010] and to reduce hash collisions [Azar *et al.*, 1999].

The compacted DBG itself is implemented with a data structure $D = (U, M)$, where U is a vector of unitigs and M is a hash table of minimizers (chapter 2.2). Each minimizer $m \in M$ is associated with a list of tuples (id_u, p_m), where id_u is a unique unitig identifier and p_m is a position in unitig id_u. The list of tuples contains the information in which

unitigs and at what starting position in the unitig the corresponding minimizer can be found. This data structure provides the important functionality to rapidly obtain the unitig for a given k-mer (e.g. when traversing the graph and searching for neighbors) by querying the minimizer of the k-mer.

Moreover, every unitig in a CCDBG is associated with a binary matrix indicating the presence or absence of each k-mer in the input sets (see chapter 2.4 for color matrix). Depending on the size and sparsity of each binary matrix different compression algorithms are applied.

Performance. In its original publication [Holley and Melsted, 2020] Bifrost built the DBG from a set $\mathcal{S}$ of 696 million short reads of the human individual *NA12878* in two hours wall clock time using 37.78 GB RAM (16 threads and k=63). 30 million short reads were queried on a DBG (16 threads and k=31) build from $\mathcal{S}$ in just over 16 minutes. The CDBG was benchmarked on up to 117,913 Salmonella strains from Enterobase[4]. Bifrost built the CDBG using 16 threads and $k = 31$ with 4000 (117,913) strains and used 1.66 (93.35) hours, 3.7 (102.74) GB RAM and no additional disc space. In comparison, the best state-of-the-art competing tool ([Muggli *et al.*, 2019] together with [Deorowicz *et al.*, 2015]) under the same conditions used 12.35 hour, 138 GB RAM and 449 GB RAM for 4000 strains and couldn't finish the task for all 117,913 strains under the given time and memory constraints.

These benchmarks showcase an unprecedented performance to store and access the genomic sequence of many individuals. In conclusion, Bifrost is a suitable DBG implementation for the demands of the software presented in chapter 4.3.

2.7 PopIns

Introduction. *PopIns* [Kehr *et al.*, 2016] is an approach to characterize insertions across a large number of individuals simultaneously. Each insertion is characterized by a genomic sequence (non-reference sequence, NRS), an insertion breakpoint on a given reference genome and the genotype of the NRS. In PopIns, determining the three properties of insertions across many individuals is divided in four distinct subproblems (Figure 14): assembly, merging, positioning and genotyping.

The *assembly* subproblem is a classical genome assembly problem as introduced in chapter 2.5. The aim of the assembly subproblem is to determine the genomic sequences that are absent from the reference genome in every distinct individual.

PopIns' approach to the assembly subproblem is to generate contigs from unmapped reads of whole-genome sequencing (WGS) data. Let $\mathcal{R}$ be the set of input short-reads. In order to generate contigs per individual PopIns' first step is to filter a set of reads $\mathcal{R}^- \subseteq \mathcal{R}$ that

[4]https://enterobase.warwick.ac.uk/

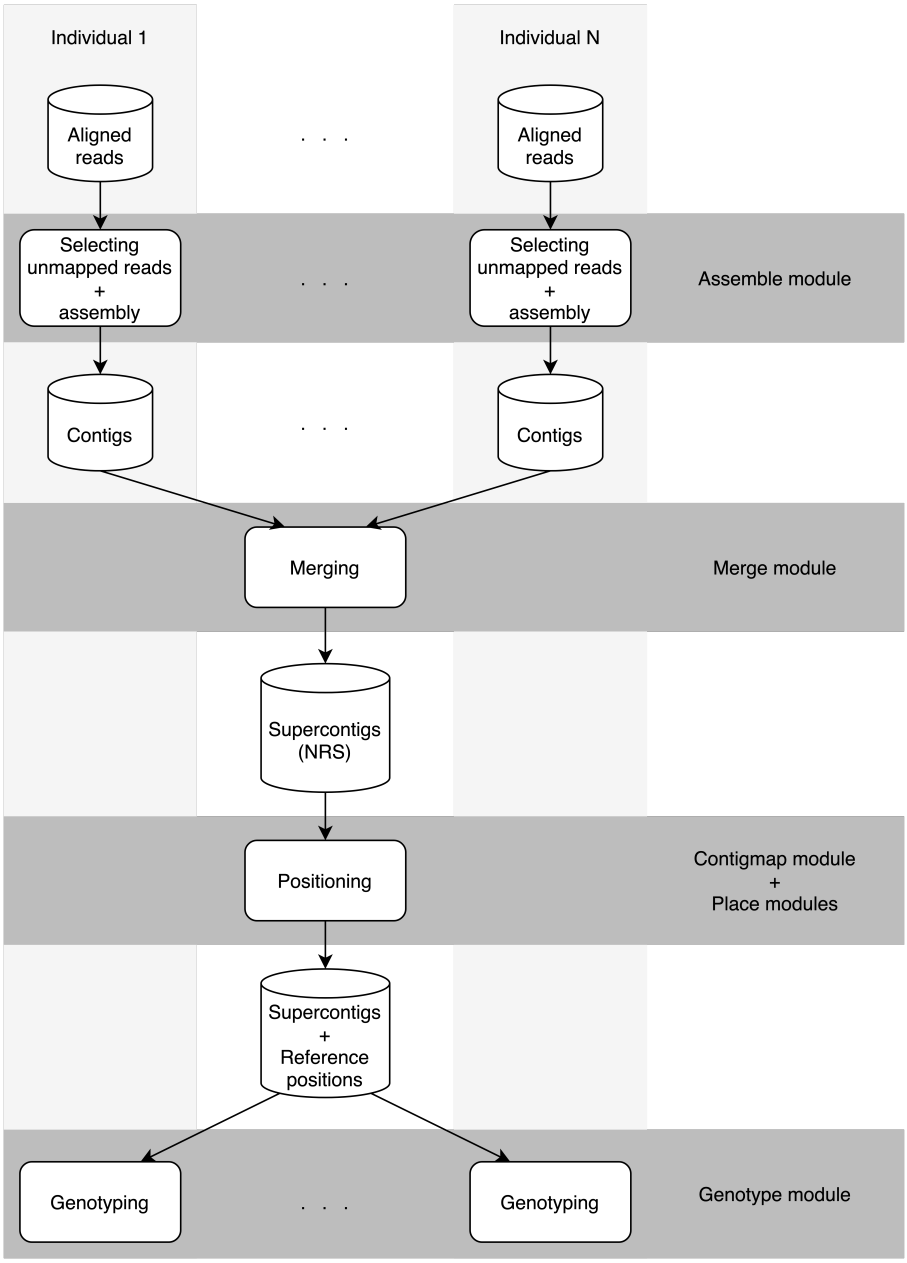

Figure 14: PopIns workflow for the detection and genotyping of NRS. For every individual unaligned reads are filtered and assembled. The resulting contigs are merged into one unified set of supercontigs. Subsequently, another step attempts to find an insertion breakpoint and orientation in the reference genome for each supercontig. Supercontigs, for which an insertion breakpoint could successfully be determined, are genotyped.

do not or only poorly align to the reference genome. Here, *poorly* is defined via a set of criteria. One important criterion is the alignment score (this will be elaborated in more detail in chapter 4.3.5). Next, each read $r \in \mathcal{R}^-$ undergoes a quality trimming [5] where fractions of r are trimmed off if their sequence does not surpass a given minimum base quality score. The remaining high quality reads are passed to an assembly tool [Zerbino and Birney, 2008] that produces a set of contigs C.

The *merging* subproblem aims to generate a set of *supercontigs* $\mathcal{S}$ from all sets of contigs $C_1, ..., C_n$, where $n > 1$, such that every $s \in \mathcal{S}$ represents an element of $\mathcal{C} = \bigcup_i C_i$ and $|\mathcal{S}|$ is minimal. A supercontig, due to its generation from the individuals' contigs, consist of a NRS and some flanking base pairs from around its insertion breakpoint. A supercontig $s \in \mathcal{S}$ is said to *represent* a contig $c \in \mathcal{C}$ if c aligns to a substring of s with a given maximum error rate ϵ, where ϵ is calculated as edit distance divided by the length of c.

In PopIns, generating the supercontigs $\mathcal{S}$ is done in two steps. The first step is to find all adjacent pairs of contigs $(c_i, c_j) \in \mathcal{C}$, where adjacent means that c_i and c_j locally align (chapter 2.2) to each other. Then, all original contigs $c \in \mathcal{C}$ are partitioned into new sets $D_1, ...D_m$. Every two contigs c_p, c_q in a set D_j, where $1 \leq j \leq m$, have to be connected by a sequence of adjacent pairs of contigs $(c_p, c_{i_1}), ..., (c_{i_r}, c_q)$ such that $(c_{i_x}, c_{i_{x+1}}) : \forall i_1 \leq i_x \leq i_r$. The second step is to generate ideally one supercontig from each D_j. In order to do this the sequence of adjacent pairs of contigs is followed and their sequence concatenated. Since the merging subproblem is a major aspect of this thesis, chapter 4.1.2 explains PopIns' merging approach in greater detail.

The *positioning* subproblem aims to determine an insertion breakpoint for every supercontig. In PopIns, this is done in two major steps (Figure 15). In the first step all read pairs of an individual where one read aligns to the reference genome and the other reads aligns to a supercontig (*anchoring read pair*) are filtered and sorted by their position in the genome. Next, a scan over the anchoring read pairs attempts to identify a cluster of reads per supercontig end that gives evidence for a potential position in the reference. Up to that point the anchoring read pairs only give a first estimate for an insertion position. In the second step all reads of an individual that have a prefix aligning to the reference genome and a suffix aligning to a supercontig or a prefix aligning to a supercontig and a suffix aligning to the reference genome (*split reads*) are filtered. The split reads are used with a split alignment (chapter 2.2) to find the precise insertion position.

In [Kehr *et al.*, 2017], a part of PopIns' original implementation that approaches the positioning subproblem was improved. If the flanking reference sequences of a supercontig align to the location (determined by anchoring read pairs) around the predicted NRS breakpoint, the verification of the breakpoint is left to the genotyping step.

[5] https://github.com/najoshi/sickle

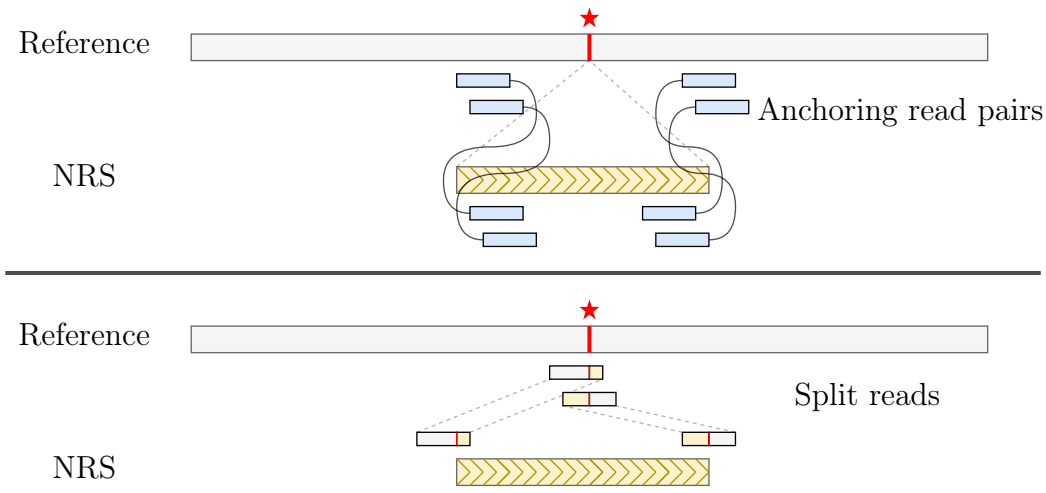

Figure 15: Sketch of the two step process how supercontigs are positioned on the reference genome. First, anchoring read pairs (blue) are used to find an approximate location for an insertion breakpoint on the reference genome. As in an anchoring read pair one read aligns to the reference and one read aligns to the NRS (upper panel), the inner distance d of the read pair provides the information that the insertion breakpoint must be within the next d bases downstream of the $3'$ end of the reference aligned read. Next, all split reads are collected that align to the genomic interval $[p+r, p+r+d]$, where p is the position on the reference genome of the $5'$ end of reference aligned read and r is the read length. The split positions of the split reads in that interval are used to determine the most likely precise position of the insertion breakpoint on the reference genome (lower panel).

The *genotyping* subproblem aims to determine the genotype of every predicted NRS breakpoint in every individual. Note that the genotyping in PopIns is tailored to diploid species, i.e. the reads $\mathcal{R}$ are presumed to originate from a diploid set of chromosomes.

PopIns approaches the genotyping subproblem by first generating the two possible alleles around each predicted NRS breakpoint. One allele R, where the NRS is absent, is just a local copy of the reference genome of ω base pairs to each side of the predicted NRS breakpoint b. The other allele A, where the NRS is present, is a concatenation of ω bases from the reference genome and ω bases from the NRS. Depending on whether the left or right end of the NRS is considered, A is either the sequence of $p-\omega$ to p concatenated with the first ω bases of the NRS or the last ω bases of the NRS concatenated with the sequence of p to $p+\omega$, respectively. Subsequently, all reads $\mathcal{R}$ of an individual are re-aligned to both alleles. From this re-alignment and its alignment scores s_S, where $S \in \{A, R\}$, the likelihoods of the NRS being present in 0, 1 or 2 chromosomal copies can be determined. The probability to observe a read $r \in \mathcal{R}$ aligned to an allele sequence S is assumed to be $P(r|S) \sim e^{s_S}$.

Hence, the likelihood of observing a read being aligned to one allele over the other is

$$P(r|S) \sim \frac{e^{s_S}}{e^{s_A} + e^{s_R}}$$

where s_A and s_R are the alignment scores of r to A and R, respectively. Computing this likelihood for every read is necessary because the quantity of reads (likely) being assigned to one or the other allele will ultimately determine the insertions genotype. Therefore, every read contributes a little evidence to the final decision of the insertion's genotype.

Under the assumption that the reads got generated from both alleles with equal probability the resulting genotype likelihoods for homozygous reference, heterozygous and homozygous alternative are

$$\begin{aligned} P(r|R,R) &\sim \frac{e^{s_R}}{e^{s_R} + e^{s_A}} \\ P(r|R,A) &\sim \frac{1}{2}\frac{e^{s_R}}{e^{s_R} + e^{s_A}} + \frac{1}{2}\frac{e^{s_A}}{e^{s_R} + e^{s_A}} = \frac{1}{2} \\ P(r|A,A) &\sim \frac{e^{s_A}}{e^{s_R} + e^{s_A}} \end{aligned}$$

respectively. Under the assumption that the reads $\mathcal{R}$ are independent the individual's genotype for a NRS is the maximum of

$$P(\mathcal{R}|S_1, S_2) \sim \prod_{r \in \mathcal{R}} P(r|S_1, S_2)$$

where $(S_1, S_2) \in \{A, R\} \times \{A, R\}$.

Again, the original implementation of PopIns that addresses the genotyping subproblem was extended in [Kehr *et al.*, 2017]. Since, in some cases, NRS were found to be flanked by duplicated reference sequence (for duplication see chapter 1, Figure 3), the genotyping considers another allele to improve the genotype prediction. Under the constraint that a precise insertion breakpoint was successfully determined, a segment of the reference sequence at the insertion breakpoint is copied and the NRS inserted between. Consequently, the allele in this model is a NRS flanked by a duplication.

Application. PopIns is a workflow of several submodules that generates NRS among the population and the individuals' insertion breakpoints and genotypes. The input at the starting point of the workflow is a read alignment (BAM and BAI files) per sample and the reference genome that was used for the alignments. The final output is a summary file (VCF file) that contains the NRS genotypes per sample. The inserted sequences are stored in a separate FASTA file. The description for how to utilize all modules can be found on the PopIns Github page[6]. The utilization of the *assemble* and *merge* module will be taken up again in chapter 5.

[6]https://github.com/bkehr/popins

Performance. In its original publication, the applicability and scalability of PopIns was demonstrated on a set of 100 simulated diploid human genomes. The entire workflow for the simulated data finished in 3 : 30h and had a hardware constraint of 16GB RAM.

PopIns was further tested on sequence data of 305 Icelandic human individuals. PopIns selected an average of 35531 unaligned reads per individual that assembled to an average of 961 contigs with a N50 of 334. The merging module reduced the total number of 210892 contigs of all individuals to 8437 supercontigs, where 6141 of them are unique to one individual.

3 Related work

This third chapter provides an overview of variant calling methods and examines the ones that yield important principles and have similar approaches to the novel method of chapter 4. First, criteria and categories are introduced to classify methods for variant calling (universal for *detection* and *genotyping*). The second subchapter provides a deeper look into the methods to detect NRS from short-read sequencing data. The third subchapter highlights the software tools that approach the NRS detection problem for many sequenced individuals simultaneously. Finally, the fourth subchapter presents recent projects that investigated structural variants in population-scale sequencing data.

3.1 Overview and classification of variant calling methods

With the rise of commercially available next-generation sequencing technologies[7,8] in 2007 the cost per sequenced genome began to decrease strongly, even outperforming *Moore's law* (Appendix, Figure 40). This rapid technological advancement initiated the sequencing of many human individuals [Levy *et al.*, 2007; Wheeler *et al.*, 2008; Schuster *et al.*, 2010] and accelerated the widespread availability of genomic sequencing data. With the available genomic sequencing data and the following algorithmic advancements [Li and Durbin, 2009, 2010; Li, 2013] in aligning sequences to a reference genome, researchers became capable to routinely identify DNA alterations (variants) in individual genomes.

Today, about a decade later, many different types of variants are know and can be distinguished (see Introduction). The detection and genotyping of different variants requires careful algorithmic design and a plethora of software tools has been developed, each of them tailored and more or less effective to detect individual variant types [Kosugi *et al.*, 2019]. The remainder of this subsection lists and elaborates on common criteria to consider when developing a method for variant calling or choosing a software tool for data analysis.

[7] https://www.roche.com/
[8] https://www.illumina.com/

Variant type. The first distinction is the variant type which is commonly defined by the length of the variant. Shorter variants up to 50 base pairs (SNPs [Sachidanandam *et al.*, 2001; International HapMap Consortium, 2003; Altshuler *et al.*, 2005] and indels [Weber *et al.*, 2002; Bhangale *et al.*, 2005; Mills *et al.*, 2006; Mullaney *et al.*, 2010]) are often considered one class as opposed to variants longer than 50 base pairs (structural variants [Iafrate *et al.*, 2004; Sebat *et al.*, 2004; Kidd *et al.*, 2008]).

Two of the most prominent variant callers for SNPs and indels are the *GATK Haplotype Caller* [Poplin *et al.*, 2018], short *GATK*, and *Freebayes* [Garrison and Marth, 2012]. Both GATK and Freebayes are computationally scalable to process many sequenced individuals.

In contrast to SNPs and indels, the length of SVs can span the majority or entirety of short reads. For instance, in [Collins *et al.*, 2020] insertions, deletion, duplications and inversions of over $100,000$ base pairs were found in multiple human individuals. Therefore, SV detection methods typically utilize signals that affect the entire read or even the read pair as evidence for a variant [Cameron *et al.*, 2019; Mahmoud *et al.*, 2019].

Signal types for SV detection. Over the course of time the myriad of tools developed to detect SVs have utilized various signals (Figure 16) from the read data:

1. *discordant alignment information*, i.e. the inner distance of a read pair differs from an expected size or one/both reads in a pair have an unexpected orientation to each other after an alignment

2. *split read alignments* (chapter 2.2), i.e. different contiguous parts of the read do not align to the reference genome or align to different parts of the reference genome

3. *coverage information*, i.e. the local read coverage of a given reference drops or increases in an unexpectedly high ratio

4. *unaligned reads*, i.e. one or both reads of a read pair have no alignment to a given reference

Early SV detection methods like BreakDancer [Chen *et al.*, 2009] used the discordant alignment information to classify read pairs into a set of normal or SV supporting reads. However, methods relying exclusively on discordant alignment information likely overlook variants of smaller size if the length of the variant is smaller or equal to the variance of the inner distance distribution of the paired-end reads [Mahmoud *et al.*, 2019]. To overcome this shortcoming other methods like DELLY [Rausch *et al.*, 2012] additionally integrated split read information. Even though small and medium size variants can be accurately detected with DELLY, larger variants were still hard to distinguish from mapping artifacts. Therefore, a third signal type, the coverage information, is used by subsequently developed SV detection tools like LUMPY [Layer *et al.*, 2014] and Manta [Chen *et al.*, 2016].

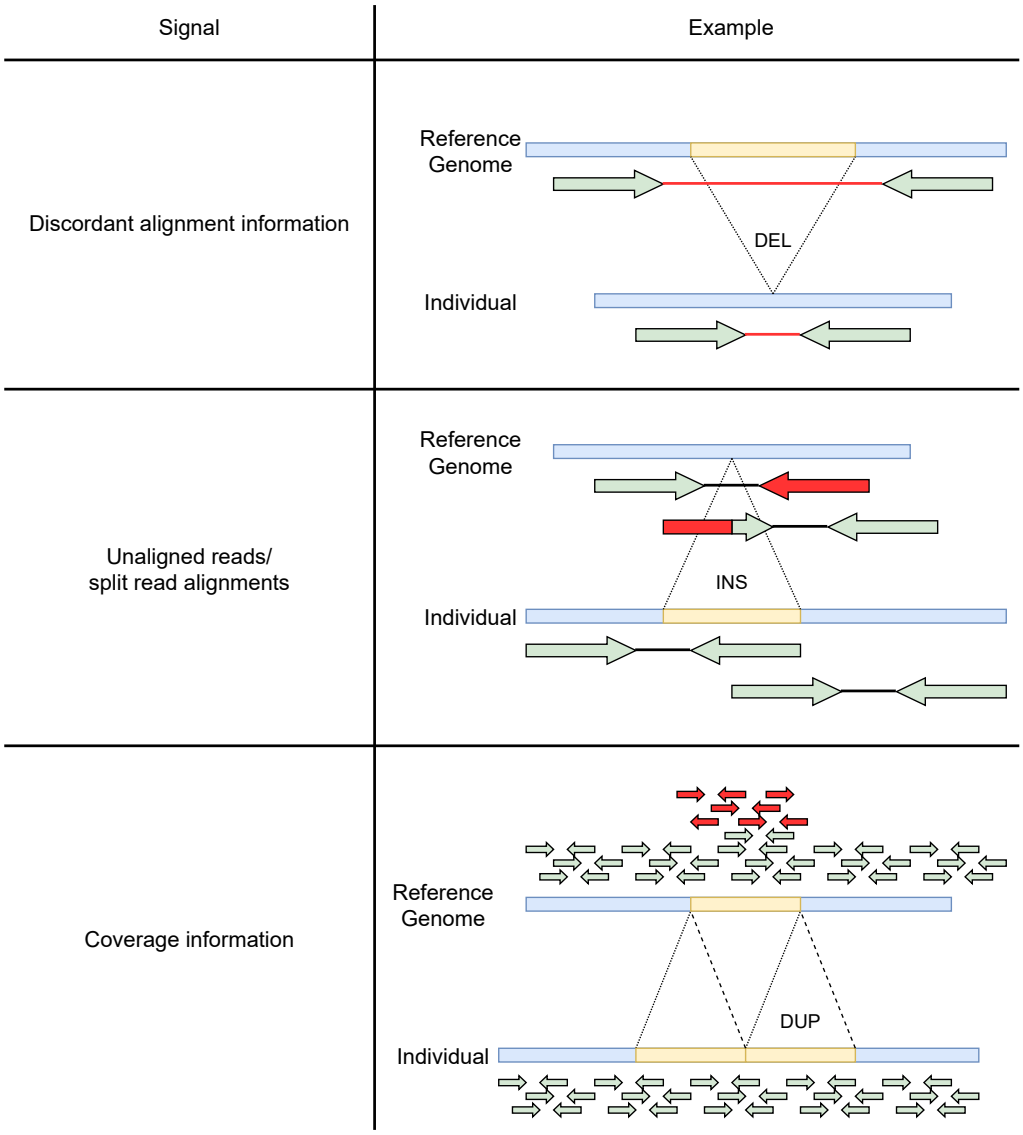

Figure 16: Examples for the different signal types which can be utilized to detect SVs. The example for discordant alignment information shows a deletion (DEL) in the individual's sequence with respect to the reference genome. The deletion causes an increased insert size (red lines) of the individual's read pair when aligned to the reference. The example for unaligned reads and split read alignments shows an insertion (INS) in the individual's sequence with respect to the reference genome. The insertion causes the individual's reads to be unaligned or only partially aligned (red arrows) when aligned to the reference. The example for coverage information shows a duplication (DUP) in the individual's sequence with respect to the reference genome. The duplication causes a locally increased read coverage (small red arrows) when the individual's reads are aligned to the reference.

Different signal types, or a combination of those, have appeared to provide better evidence for individual SV types. Deletions can be detected with a variety of signals like discordant alignment information, coverage and split reads if the deletions are small. The vast majority of methods to detect *copy number variants* (CNV), e.g. tandem or interspersed duplications, is using read depth information [Zhao *et al.*, 2013]. Methods to detect insertions (the chapters 3.2 and 3.3 discuss NRS detection in greater detail) typically utilize split read alignments and unaligned reads [Cameron *et al.*, 2017; Ye *et al.*, 2009].

As some signals, and hence some methods, turned out to be favourable to detect particular variant types so called *meta methods* [Jeffares *et al.*, 2017; Mohiyuddin *et al.*, 2015; English *et al.*, 2015; Zarate *et al.*, 2020; Fang *et al.*, 2018] have been developed that combine multiple SV detection methods to obtain a comprehensive callset comprising all variant types.

Sequencing technology. The aforementioned methods all have in common that they operate on plain short-read whole-genome sequencing data. That implies certain constraints, expectations and properties of the reads, e.g. read length, per base error model, random but nearly uniform sampling across the genome, pairing of reads and coverage. In the era of sequencing after the initial next-generation sequencing technologies the rise of novel sequencing technologies and protocols (Pacific Biosciences CLR[9], Pacific Biosciences CCS/HiFi[10], 10X Genomics Chromium Genome and Exome[11]) requires novel algorithms and software tailored to their new formats of reads.

In brief, the third-generation sequencing methods by Pacific Biosciences or Oxford Nanopore[12] generate much longer and unpaired reads [Amarasinghe *et al.*, 2020; Ou *et al.*, 2020], hereafter just called *long reads*. Structural variant detection and genotyping methods using long reads commonly utilize two beneficial properties of the reads:

1. Long reads typically have much longer alignments with the reference that is not affected by the variant than short-reads.

2. The sequence length of long reads can span a much larger size spectrum of structural variants than short-reads.

Sniffles [Sedlazeck *et al.*, 2018] and SVIM [Heller and Vingron, 2019] are two SV detection methods which utilize at least one of (1) and (2). The signals are utilized to either cluster breakpoints of the split read alignments or explicitly cluster genomic sequences which deviate from the reference genome.

[9] https://www.pacb.com/tag/continuous-long-read-clr/

[10] https://www.pacb.com/smrt-science/smrt-sequencing/hifi-reads-for-highly-accurate-long-read-sequencing/

[11] https://www.10xgenomics.com/blog/a-basic-introduction-to-linked-reads

[12] https://nanoporetech.com/

Read alignment vs genome-to-genome alignment. All the aforementioned methods investigate structural variants based on the alignment of reads to a reference genome. For the sake of completeness it should be mentioned that the routine assembly of human genomes has become increasingly feasible over the last years [Li, 2016; Ebert *et al.*, 2021] and that novel SV detection methods [Li, 2016; Heller and Vingron, 2020] can utilize assembled sequences for a comparison of sample and reference.

Sample size. Another increasingly important aspect for variant callers is the number of samples the methods can process and whether the methods can make use of the data from multiple individuals. Early SV detection methods were primarily designed to operate with the data of one single individual. It was shown [Kehr *et al.*, 2016; Niehus *et al.*, 2021] that most of these methods have a restrictive scalability of computational resource requirements or a notable drop in their prediction accuracy with an increasing number of individuals given to the programs. To circumvent the former there are methods [Kirsche *et al.*, 2021; Eggertsson *et al.*, 2019; Jeffares *et al.*, 2017; Becker *et al.*, 2018] to merge the variant callsets from many individuals.

However, *a posteriori* merging of variant callsets from many individuals does not utilize the explanatory power of population-scale data. This way weak signals or incomplete data at the scope of a single individual is likely to vanish in the mass of data (more details in chapter 3.3). Contrary to methods that are designed for the use on only single individuals other methods like SVIM or pbsv[13] can process multiple individuals at once and leverage their joined signals. SVIM can seamlessly incorporate and maintain the alignment signals from the long reads of many individuals. Only the many pairwise computations of its *span-position distance* together with the *maximal clique finding* algorithm constrain the number of individuals that can be processes at once. The precise details and source code of pbsv are undisclosed intellectual property of Pacific Biosciences but the program is designed to process multiple individuals. Another program [Niehus *et al.*, 2021] to detect and genotype structural variants jointly from the short-read data of up to thousands of individuals is PopDel. PopDel utilizes discordant alignment information (deviations from the *insert size*, which is related to the inner distance of Figure 6) to detect SVs. In the variant calling phase of PopDel the signals of all samples are used for a likelihood ratio test per given window size of the reference genome.

A striking conceptual similarity in the design of pbsv and PopDel is, even though they operate on different sequencing technologies, that both programs first gather and compress the signals for every individual before they deploy their algorithms to validate the signals jointly from all individuals. This is a common pattern in algorithms that operate on population-scale data and is implemented, in a different form, in methods for NRS detection too (chapters 3.3 and 4).

[13] https://github.com/PacificBiosciences/pbsv

Germline vs somatic variants. Finally, many methods have a clear distinction whether they examine germline or somatic variants [Gong *et al.*, 2021]. To understand their potential similarities and differences[14] from a broad perspective three aspects will be highlighted here using the tools GATK HaplotypeCaller and Mutect2 [Benjamin *et al.*, 2019].

The first aspect and difference is the samples that are compared. In germline variant calling a *normal* sample (blood, tissue, saliva, etc.) is compared to a *control* (e.g. reference genome, other individuals from the same pedigree). In contrast, somatic variant calling works preferentially with a pair of *case* (tumor) and normal sample. Mutect2 contrasts a case and normal sample from the same individual and by its definition only variants that differ from the normal and reference are reported as somatic. The second aspect is the detection of variants or their signals from the data. A method to detect a particular variant type can usually be applied to detect either somatic or germline variants as they exhibit the same signal type. For instance, a deletion will always cause a greater inner distance of a read pair independent of whether the variant is somatic or germline. The third aspect is the genotyping of the variant. In germline variant calling the genotype likelihoods and genotypes are typically calculated under the assumption of a predefined fixed ploidy [Kehr *et al.*, 2016; Poplin *et al.*, 2018]. In contrast, somatic variant calling as in Mutect2 is not constrained to a fixed ploidy and can report any fraction of allele frequency. This extra degree of freedom in somatic variant calling is important as tumor samples can have varying ploidy due to the fractional sample purity, different subclones or CNVs.

3.2 Methods for detection and genotyping of non-reference sequences

Only few of the aforementioned methods in chapter 3.1 are capable to detect and genotype insertions, even less are capable to do so using short-read sequencing data. Insertions in particular are a less frequently investigated class of structural variants because as opposed to other classes of structural variants the full description of an insertion requires not just the breakpoint on the reference genome but also the genomic sequence (NRS) that is missing from the reference. If the NRS is longer than the read, which is often the case for short-read data, a local sequence assembly of unaligned reads into the NRS is inevitable. Therefore, many software tools for SV discovery omit insertion detection as sequence assembly adds a high algorithmic complexity.

[14] https://gatk.broadinstitute.org/hc/en-us/articles/360035890491-Somatic-calling-is-NOT-simply-a-difference-between-two-callsets

Methods using short-reads for insertion detection comprise of three major steps:

- Local or whole-genome assembly of NRS
- Determining the position of each NRS in the reference genome
- (Optional) Genotyping of the detected variant

The remainder of this subsection outlines how different methods approach these step.

Two early methods for the assembly and breakpoint detection of NRS are Pindel and SOAPindel [Ye *et al.*, 2009; Li *et al.*, 2013]. Both approaches filter anchoring read pairs. Pindel determines an *anchor point* on the reference genome with the aligned read, i.e. knowing the position and orientation of the aligned read in the reference genome constraints the expected alignment location for the unaligned other read in the pair. The unaligned read is split into three fragments and the two terminal fragments are separately re-aligned to the reference within the expected interval. This approach restrains the NRS detection to small (smaller than a read) variants. Thus, its succeeding method SOAPindel added a local assembly step to detect larger NRS (larger than a read). SOAPindel piles up unaligned reads at their expected *virtual* genomic location and identifies cluster. If there are enough unaligned reads in such a cluster the reads are assembled into a NRS. Mind that both Pindel and SOAPindel rely on one read in a pair being accurately aligned to the reference genome.

About the same time another method, NovelSeq [Hajirasouliha *et al.*, 2010], followed a different approach to solve the detection also for larger NRS. One major difference to Pindel/SOAPindel is in the first step of NovelSeq where it filters not just read pairs with precisely one unaligned read in the pair (in the original publication called *one-end anchored* reads, short *OEA reads*) but also reads pairs which have no aligned read (in the original publication called *orphan* reads). The orphan reads are assembled into *orphan contigs* using external assemblers like EULER [Chaisson and Pevzner, 2008] or ABySS [Simpson *et al.*, 2009]. In a separate process OEA reads around insertion breakpoints on the reference genome are clustered and assembled into *OEA contigs*. Finally, a merging process attempts to align an OEA contig to each side of an orphan contig. NovelSeq gained recognition as NRS detection method in the pilot phase of the 1000 Genomes Project [Mills *et al.*, 2011]. Unfortunately, a functional version of the tool was never made available.

A few years after the development and deployment of the alignment-based NRS detection methods a new tool, MindTheGap [Rizk *et al.*, 2014], was developed. MindTheGap approaches the assembly and breakpoint detection step from a different angle than the other tools before. Instead of using pre-computed alignments of the reads MindTheGap operates on k-mers, in particular on the difference of k-mer sets. At first, MindTheGap builds a de Bruijn Graph from all reads. Next, a comparison to the k-mers present in the reference genome identifies runs of k-mers in the graph that are absent from the reference.

The transitions of present and absent k-mers in the graph are treated as putative NRS breakpoints in the reference genome. The second step is the assembly of NRS from the graph. For each NRS breakpoint of the first step a (L, R)-pair of k-mers is extracted that flanks the breakpoint from the left and right on the reference genome, respectively. The L k-mer is used as start node for a BFS through the graph component of k-mers that are absent from the reference genome. The BFS goes on (under some maximum branching constraints [Chikhi and Rizk, 2013] until the R k-mer is found in a node of the graph. Finally, the path of k-mers from L to R in the dBG is converted to a consecutive sequence, the NRS.

BASIL and ANISE [Holtgrewe *et al.*, 2015] is a NRS detection method that extends the ideas of [Hajirasouliha *et al.*, 2010] for insertion site detection and implements an assembly step (ANISE) with a technique to better resolve the assembly of highly repetitive sequences. Once the OLC assembly has created an initial contig the corresponding reads are re-mapped to the contig to identify base positions that differ from the contig. Next, a statistical test checks if a significant number of reads exhibits differences at common base positions in the contig. A cluster of reads that passes the test is then used to assemble another copy of the location they were taken from. The newly assembled copies vary by few but statistically verified base positions. As a finalizing step ANISE uses a *digraph-based scaffolding* procedure to bring an order into the new copies.

3.3 Methods for detection and genotyping of non-reference sequences in population-scale data

All methods for NRS detection in chapter 3.2 were designed for the application of one donor genome at a time. None of the methods has an algorithmic strategy that is particularly designed to leverage short-read sequence data from many individuals. However, since the commercial availability of high-throughput whole-genome sequencing initiated an era of population-scale sequencing it is evident that the genomics community needs variant calling methods that scale to large sample sizes. As explained earlier, SV detection methods which consider NRS variants are comparatively rare and even fewer exist for population-scale data. This subsection summarizes the two methods, Cortex [Iqbal *et al.*, 2012] and Pamir [Kavak *et al.*, 2017], that compare the closest to the original PopIns and the method described in chapter 4. Both methods are tailored to short-read data, can process multiple individuals simultaneously and can detect NRS variants.

Cortex is a tool for metagenome assembly, i.e. it attempts to assemble entire strains of microbes, and is able to call variants from its graph structures. Initially, Cortex builds a DBG for every sample using multiple sizes of k. Next, the individual DBGs are merged into a colored DBG. The colored DBG can be built from at least one sample and a reference genome or at least two samples if built without a reference genome. Depending on the presence or absence of a reference genome Cortex uses the *Path-Divergence* or *BubbleCalling* algorithm [Iqbal *et al.*, 2012], respectively, to detect and

genotype variants based on the different paths that alternative alleles induce in the CDBG.

Pamir is inspired by many of the principles of NovelSeq but was designed to process many individuals simultaneously. Similar to NovelSeq, Pamir filters orphan reads and OEA reads first and assembles the orphan reads to orphan contigs. But in contrast to NovelSeq Pamir does not immediately assemble the OEA reads. The OEA reads that are not aligned to the reference or only have a partial alignment are grouped according to the location of their mapped counterpart. Next, Pamir generates clusters, each consisting of a group of OEA reads and one or multiple orphan contigs that align to the OEA reads. The clusters are assembled to *insertion candidates* and locally aligned to a confined reference region (determined by the aligned OEA reads). Finally, insertion candidates with a successfully determined insertion breakpoint are genotyped similar to the genotyping approach described in 2.7.

The novel algorithmic feature that enables Pamir to leverage the data from multiple individuals is that the assembly of the orphan contigs and the formation of the clusters can recruit the read pairs from multiple given individuals. As a result, the set of insertion candidates contains the NRS sequences from all individuals. All insertion candidates in that joint set have their genotype determined for every individual.

In summary, the colored de Bruijn Graph and Cortex' algorithms on CDBGs have been milestones in the analysis and comparison of genomic sequences across individuals. However, the implementation of the CDBG and Cortex at the time of its original publication was designed for bacterial genomes not applicable to a great number of large and highly complex genomes. PopIns is one of the first approaches for the joint detection of NRS across multiple individuals that was demonstrated to work on (unaligned) human sequencing data and to work effectively on data sets of several hundred individuals. Later, Pamir demonstrated a superior recall to PopIns in some scenarios but was never successfully applied to data sets whose cardinality match those processed by PopIns.

3.4 Selected projects conducting variant calling using population-scale data

This last subsection provides an overview of selected projects which deployed NRS variant calling at population-scale.

Country-specific sequencing projects. Over the past few years, national cohorts have become a rising trend in genomics. Multiple (mostly Western and Asian) countries started or are already actively sequencing great quantities of individuals [Hehir-Kwa *et al.*, 2016; Gudbjartsson *et al.*, 2015*a*; Eisfeldt *et al.*, 2020; Duan *et al.*, 2019; Maretty *et al.*, 2017; Sherman *et al.*, 2019].

In 2016, a Dutch human reference panel [Hehir-Kwa *et al.*, 2016] of 769 individuals was used to detect an overall 1.9 million SNPs and SVs. Among those variants it was over 4 megabases of novel sequence detected. 191 known and trait-associated SNPs were found to be strongly related to the presence of SVs. For the structural variant discovery 12 different tools have been used (with Pindel among the ones mentioned in 3.2) which together comprise all different signal types.

Similarly, in 2019, reports of the Human Pan-genome Analysis were published [Duan *et al.*, 2019] comprising the variant calling of 275 Han Chinese genomes. Here, a total of 29.5 megabases of novel genomic sequence was detected with at least 188 novel protein-coding genes.

One of the largest genomic data resource of a single human population is collected on Iceland [Gudbjartsson *et al.*, 2015*a*; Jónsson *et al.*, 2017] comprising the whole genome sequences of about 60,000 individuals[15]. In their studies [Gudbjartsson *et al.*, 2015*b*; Jónsson *et al.*, 2017; Kehr *et al.*, 2017; Beyter *et al.*, 2021] about sequence diversity and structural variants they recently discovered a median 22,636 SVs (median of 13,353 insertions) per genome. Structural variant callsets at various stages of data collection have been used successfully to identify high risk variants, e.g. causing myocardial infarction [Kehr *et al.*, 2017] or abnormal cholesterol levels [Bjornsson *et al.*, 2021].

Sequencing projects of diverse human populations. Aside from the projects investigating structural variants in a single human population there is a growing number of large-scale sequencing project comprising individuals of various ancestries [McVean *et al.*, 2012; Auton *et al.*, 2015; Telenti *et al.*, 2016; Mallick *et al.*, 2016; Wong *et al.*, 2018, 2020; Taliun *et al.*, 2021; Byrska-Bishop *et al.*, 2021].

One of earliest and most prominent diverse population-scale sequencing project is the 1KGP. With initially 1,092 [McVean *et al.*, 2012] and by now 3,202 human genomes [Byrska-Bishop *et al.*, 2021] it remains one of largest publically available resources for human sequence data of various sequencing technologies and pre-computed variant callsets. Early investigations [Mills *et al.*, 2011; McVean *et al.*, 2012] of SVs in the 1KGP were widely limited to deletions. With the latest phase 3 release of the 1KGP more SVs types were investigated [Auton *et al.*, 2015; Sudmant *et al.*, 2015] but insertion detection was still limited to Alu and L1 elements. A latest work [Byrska-Bishop *et al.*, 2021] investigated all fundamental types of SVs (Figure 3) with short-reads aligned to a more recent version (GRCh38) of the human reference genome and a higher per-sample read coverage (mean depth 30x instead of 7.4x) than earlier work. The ensemble callset of all individuals comprises 51,829 insertions that were curated and integrated from three different pipeline. Mind that these insertions do not necessarily comply with this thesis' definition of strictly being absent from the reference genome. In general, the quantity of SVs in the ensemble callset is 170% higher than in the phase3 release.

[15] https://www.the-scientist.com/profile/master-decoder–a-profile-of-kri-stefnsson-65517

4 Methods

The method described in this chapter was developed and evaluated in collaboration with W. Timothy J. White, Sebastian Niehus, Birte Kehr, Guillaume Holley and Bjarni V. Halldórsson. Birte Kehr supervised the research project. The method was presented at scientific conferences and seminars including Genome Informatics 2018, Computational Genomics Summer Institute (CGSI) 2019, European Society of Human Genetics (ESHG) 2020 and RECOMB Satellite Workshop on Massively Parallel Sequencing (RECOMB-Seq) 2021, and published in Bioinformatics by Oxford University Press:

Krannich T., White W.T.J., Niehus S., Holley G., Halldórsson B.V., Kehr B. Population-scale detection of non-reference sequence variants using colored de Bruijn Graphs. Bioinformatics 2021, btab749, https://doi.org/10.1093/bioinformatics/btab749

This chapter proposes a novel algorithm [Krannich *et al.*, 2021] for the detection and genotyping of long sequence variants that are absent from the reference genome but present in one or many individuals. The first subchapter examines limitations of its predecessor, classifies the new method in terms of the criteria in chapter 3.1 and provides an informal description of its objective. The second subchapter introduces a precise, formal description of the new method and its core mechanism is formulated as an optimization problem. Moreover, a greedy heuristic approach to find an approximate solution is introduced. The third and last subchapter elaborates on the implementation of the new method in more detail, highlights particular features of the software and describes its best-practise usage.

4.1 The roadmap of Popins2

4.1.1 Motivation

Building on top of Popins. The starting point for this work was PopIns ([Kehr *et al.*, 2016], chapter 2.7), an algorithm and software developed to detect and genotype NRS variants from population-scale NGS data. A major driver that led to the design principles

of PopIns is the observation [Miller *et al.*, 2010; Zerbino *et al.*, 2012] that the assembly problem (chapter 2.5) for a single individual requires high-coverage sequencing data. As the detection of large NRS variants inevitably requires sequence assembly, NRS callsets derived from low-coverage data are often fragmented and largely incomplete [Alkan *et al.*, 2011]. This is a particularly hard problem for the detection of NRS variants with a low-frequency in a given study group [Kehr *et al.*, 2016]. For instance, large projects that were instigated to include, but are not limited to, the detection of rare diseases might overlook disease causing variants or underestimate allele frequencies.

Popins was designed to overcome such limitations. Its merge step joins the sets of contigs from unaligned reads of an entire study group and thereby increases the total sequence coverage of NRS that are common in multiple individuals. An increasing number of individuals added to this merging strategy increases the chance that the contigs from multiple individuals compensate the uncovered bases in the NRS of a single individuals. As a result, the NRS detected across multiple individuals are typically less fragmented and enable genotyping even in individuals where the read coverage is not sufficient to assemble a particular NRS.

In summary, PopIns is a carefully designed insertion detection algorithm whose submodules seamlessly integrate into a comprehensive workflow. PopIns' merge step jointly uses the signal from many individuals during the variant detection.

Scalability for larger cohorts. A major challenge for population-scale insertion detection methods like PopIns is the processing and maintenance of huge amounts of data. An increasing amount of data added to the problem instance affects the detected sequences, duration of its computation and memory consumption.

The aforementioned advantages of PopIns may suggest that more data automatically leads to more accurately detected NRS. However, in practise it is challenging to retain the signal-to-noise ratio, i.e. with more sequence data added to the method it gets increasingly difficult to not amplify sequencing artifact as well. Moreover, many biological sequences (especially from eukaryotic cells) contain various repetitive elements [Jurka *et al.*, 2005, 2007; Chen, 2004] that are difficult to assemble. The complexity to accurately reconstruct the original biological sequence of these repetitive elements increases with a growing number of individuals.

PopIns is a modular software that can process sequence data at various stage of the insertion detection with dedicated units of the program. However, particularly the merge step is a bottleneck in the computational performance of PopIns as it inevitably needs to handle the data of the entire population or study group. At its core, the merge step of PopIns groups the many contigs from all individuals by sequence similarity [Rasmussen *et al.*, 2006] and assembles the sequences of each group via an approach similar to [Feng and Doolittle, 1987]. The determination of sequence similarity and its subsequent assembly

of all contigs across the entire population is a dominating if not limiting factor for the scalability of PopIns. The time of execution and memory consumption rapidly increase with a growing number of individuals.

In summary, detecting insertions at population-scale is challenging in terms of an accurate NRS assembly and computational resource management.

CDBG as replacement. In the past, assembly via DBGs [Chaisson and Pevzner, 2008; Bankevich *et al.*, 2012; Zerbino and Birney, 2008; Li *et al.*, 2015; Chikhi and Rizk, 2013] has become a frequently used alternative approach for the OLC paragigm [Kececioglu and Myers, 1995; Sutton *et al.*, 1995; Miller *et al.*, 2010] for the assembly of NGS data. The DBG has the advantage that many established graph algorithms in computer science can be applied to it as every DBG can be considered to be a directed graph (or undirected graph if it is a bidirectional DBG, see chapter 2.4). For instance, the EULER [Pevzner *et al.*, 2001] and Minia [Chikhi and Rizk, 2013] assemblers implemented an Eulerian path [Euler, 1736] and breadth-first search approach on a DBG. Also, the CDBG has successfully been used for the *meta-genomic* analysis of many bacterial strains [Iqbal *et al.*, 2012, 2013]. If implemented carefully [Holley and Melsted, 2020; Khan and Patro, 2021; Khan *et al.*, 2021], the DBG and CDBG have a low memory footprint.

In summary, the DBG and CDBG are entrenched data structures for assembly and sample comparison. DBGs scale particularly well if the many added sequences are highly similar.

Synopsis. PopIns introduced a sophisticated method and a practical implementation that, for its purpose, scales to an unmatched numbers of individuals. Especially its merge step comprises an elaborate concept to merge many highly similar contigs. Nevertheless, the scalability in the application of PopIns [Kehr *et al.*, 2017] has been limited by the algorithmic complexity and massive compute resource consumption of the merge step. The CDBG is a practical data structure that can be used as a foundation to rework the merge step. These insights, together with a continuously growing number of individuals in population-scale projects, motivated the development of a successor of PopIns that detects NRS from the same set of individuals using much less compute resources and with at least equivalent accuracy.

4.1.2 Objective

This subchapter explains which methods and data structures of PopIns are subject to changes and which novel contributions have to be done in order to achieve a scalability to greater numbers of individuals. It is advised to have read the chapters 2.7 and 4.1.1 for the fundamental preliminaries of PopIns merge and the motivation to develop this new approach, respectively.

The approach of PopIns2 to jointly process more individuals than its predecessor is to replace the old data structures of the merge step with a CDBG and to develop an algorithm that generates a set of supercontigs $\mathcal{S}$ from the CDBG.

Replacing data structures. PopIns utilizes two data structures in the merge step (Figure 17) to implement an assembly procedure similar to OLC. One data structure is a union-find data structure that maintains sets $C_1, ..., C_n$ of similar contigs. This data structure has two major drawbacks. It reallocates and doubles the size of compute memory every time a set cannot store another contig because it reaches its maximum capacity. Further, within a set C_i there is hardly any compression or reduction in redundancy of similar sequences. The second data structure is a sequence graph for every C_i, where vertices store subsequences of the contigs $c \in C_i$ and directed edges represent adjacencies between a pair of substrings. A depth-first search through a sequence graph is used to generate supercontigs, representative for the consensus phase of the OLC paradigm. However, if there are many branching components in the graph, the time complexity of this approach can become prohibitive [Kehr *et al.*, 2016]. A new approach is to replace both data structures from PopIns' merge step with a CDBG.

Multi-sample local assembly. Given the new data structure, the CDBG, it requires a novel algorithmic method to generate a set $\mathcal{S}$ of supercontigs from the graph that represents a non-redundant set of NRS from an entire population of given individuals. The objective is to implement an assembly approach (chapter 2.5) that builds a CDBG from the contigs of many individuals, generates a set of paths $\mathcal{P}$ from the CDBG (see chapter 2.4 for paths) and finally translates the paths into supercontigs. Intuitively, a set $\mathcal{P}$ of paths from the CDBG $G = (V, E, C)$ has to fulfill three properties:

1. Each path $p \in \mathcal{P}$ should have a high conformity of colors between any two consecutive vertices u_i, u_{i+1}, where $u_i, u_{i+i} \in p$.

2. Every $v \in V$ should be a vertex in at least one path $p \in \mathcal{P}$.

3. $\mathcal{P}$ should have low redundancy δ, i.e. any two path $p_i, p_j \in \mathcal{P}$ should have a smallest possible sum
$$\delta = \sum_{\substack{u \in p_i \\ w \in p_j}} u == w,$$
where '==' is the logical equivalence operator that returns 1 if u and w are the same vertex in V or 0 otherwise.

Property (1) is the property that identifies similar or equivalent contigs of across individuals. In other words, paths from the CDBG that have a long streak of vertices with similar colors likely result from similar contigs. Property (2) aims for the data integrity from input (contigs) to output (supercontigs) for the approach. The necessity for this property can be outlined with a short counter proof. A basic assumption is that every contig that goes into the CDBG is already a NRS or part of a NRS. Therefore, missing a vertex $v \in V$

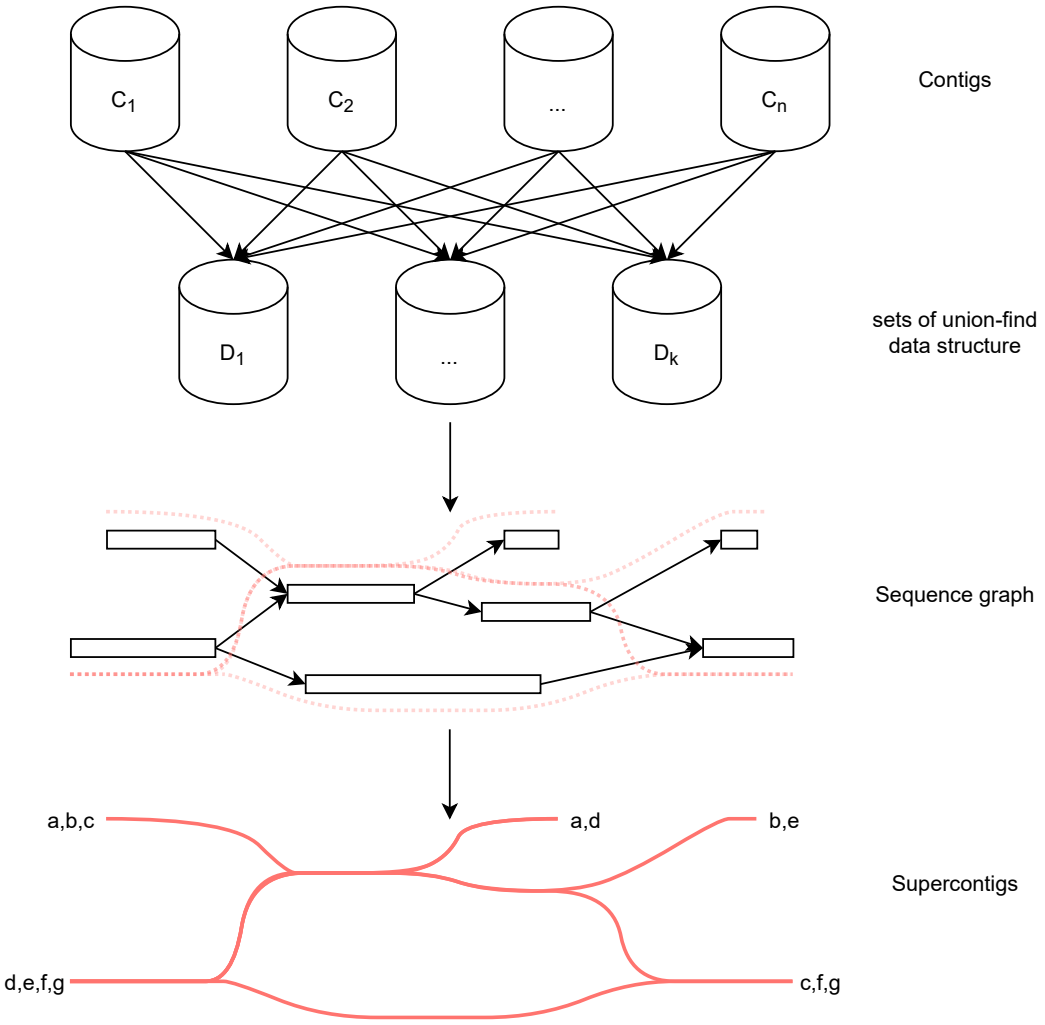

Figure 17: Merging method of PopIns. The sets $C_1, ..., C_n$ are contig sets from n individuals. Let $\mathcal{D} = D_1, ..., D_k$ be a union-find structure where each $D \in \mathcal{D}$ is a set of contigs. A contig $c \in C$ is added to $\mathcal{D}$ by taking the union of all sets $D \in \mathcal{D}$ that contain a contig that aligns to c and finally adding c to that union. If c has no alignment to the contigs of any $D \in \mathcal{D}$ then a new set is added containing only c. Next, a sequence graph $G = (V, E)$ is built for each set D, where V are substrings of contigs and E are adjacencies between the substrings. In order to add a contig $d \in D$ to G all paths in G are enumerated (red dashed lines), d is locally aligned to the concatenation of substrings of all paths and the path with the best alignment is chosen. Depending on the position of the local alignment to the path, nodes are split and added or a prefix or suffix is added to a label of a leaf node. Finally, a concatenation of all paths starting and ending in a leaf node (solid red lines) are reported as supercontigs. The identifiers a,b,c,d,e,f,g indicate the start and endpoints of all paths that make a supercontig.

from $\mathcal{P}$ means that (parts of) a NRS is missing from the supercontigs. This is contrary to the objective. Finally, property (3) determines the redundancy of the supercontigs. If the approach would enforce $\delta = 0$ it means that all paths in $\mathcal{P}$ are *pairwise disjoint*. However, two different NRS might share one or multiple k-mers and therefore the corresponding two paths need to contain the same vertex of the shared k-mer. If the approach allows for $\delta > 0$ it means that any two paths $p_i, p_j \in \mathcal{P}$ can share one or more common $v \in V$. In a more extreme scenario, if δ is a very large integer the approach might generates two equal paths (both paths have the same or reverse sequence of vertices) or one path is a subsequence of the other. The sought approach requires a way to control and find a sensible setting for the redundancy.

4.1.3 Classification

Chapter 3.1 gave an overview of different criteria to consider for the development or application of NRS detection methods. Here, the method's properties and intended domain of application of the second major version of PopIns, *PopIns2* [Krannich *et al.*, 2021], are defined in terms of those criteria.

- **Variant type.** PopIns2 is designed for the sequence assembly, breakpoint detection and genotyping of NRS. While chapter 1 introduced SV as a set of variants typically larger than 50 bp, the minimum length of NRS that can be detected with PopIns2 is $2k - 1$.

- **Signal type.** From all reads aligned to the reference PopIns2 filters unaligned reads and partially aligned reads. Unaligned reads are utilized to assemble the contigs that ultimately result in NRS and to associate supercontigs (chapter 2.7) with approximate breakpoint positions on the reference genome. Poorly aligned reads are utilized for a split read alignment. Successful split read alignments are utilized to determine the precise base pair position of the insertion breakpoint in the reference genome.

- **Sequencing technology.** PopIns2 is tailored to short-read paired-end sequencing technologies. It was designed, implemented and tested exclusively with Illumina short-read WGS data.

- **Alignment type.** PopIns2 utilizes the information of read alignments. Reads are either aligned to each other, to supercontigs or to the reference genome. At no stage of the algorithm whole genomes are assembled or aligned.

- **Sample size.** PopIns2 is designed to process the NGS data of hundreds to thousands of individuals. PopIns2 jointly utilizes the data of many individuals during NRS detection.

- **Germline and somatic variants.** The NRS detection phase of PopIns2 is applicable to investigate germline as well as somatic structural variants. However, the genotyping phase assumes the reads to originate from a diploid genome. This assumption might be violated for somatic cells.

None of these classifications changed with the advancement from PopIns to PopIns2.

4.2 Merging NRS of many genomes using a CDBG

While chapter 4.1.2 aimed to explain the objective of PopIns2's new merging procedure in an intuitive way, this chapter presents a formal problem definition and provides a greedy approach to solve the problem.

4.2.1 Problem formulation

Derivation. The objective of the new merging step is to extract NRS from a CDBG that was created from the sets of contigs generated by the assembly step (chapter 2.7) applied to the reads of each individual. This means that, in contrast to whole-genome assembly which aims to minimize the amount of contigs and maximize the contiguity (see chapter 2.5 at Evaluation statistics), the new merging approach aims to generate many comparably short genomic sequences. The assembly of many short NRS was found to be more related to transcript assembly [Xing *et al.*, 2004; Trapnell *et al.*, 2010] than to whole-genome assembly. An important difference is that transcript assembly algorithms typically operate on a directed acyclic graph (DAG). Similarly to a transcript assembly, the sought set of paths through the CDBG has to be a path cover (see chapter 2.4) of the CDBG assuming that every vertex (k-mer or unitig) is part of at least one NRS. The rationale for this condition to be desired was discussed in the previous chapter. A naive solution to generate a path cover $\mathcal{P}$ is to enumerate all possible paths in the CDBG. However, this solution comprises a large amount of paths with widely duplicated subsequences of vertices and therefore leads to a large and unwanted redundancy. Also, many of the paths would not correspond to actual NRS.

The weighted minimum path cover problem. The problem of the new merge step is formulated as a *minimum path cover problem*. That is to find the smallest number of paths in a given graph that forms a path cover. The minimum path cover problem is NP-complete [Garey and Johnson, 1990] for all graphs other than DAGs [Lawler, 2001]. It was shown [Rizzi *et al.*, 2014] that the smallest solution to test, namely whether there exists a path cover of cardinality one, is equivalent to finding a Hamiltonian path. In a minimum path cover some paths can share one or many common vertices. The biological equivalent of these common vertices from a CDBG is shared genomic sequence between multiple individuals, e.g. low complexity sequence or mobile elements.

If applied to a CDBG, the minimum path cover problem in its original form has no constraint that utilizes the information of the underlying contigs, yet. Therefore, the

problem formulation is extended by weights of the paths. The idea is that if multiple genomes carry the same or a similar NRS there will be a path in the CDBG with the corresponding colors of the genomes. More precise, all k-mers along a path that corresponds to an actual NRS should be labeled with a similar set of colors. Using the color vectors of the k-mers and the Jaccard index for bitvectors J (chapter 2.1) the weight function $\phi : p \rightarrow [0, 1]$ that assigns a weight $\phi(p)$ to each path $p = u_1, u_2, ..., u_n$ through a compacted CDBG is defined as

$$\phi(p) = 1 - \min_{1<i<n-1} J(\text{last}(v_i), \text{first}(v_{i+1})) \tag{1}$$

where $last(v_i)$ and $first(v_{i+1})$ are the bitvectors corresponding to the last and first k-mer of vertex (unitig) v_i and v_{i+1}, respectively. Using Equation 1 for the path weights, the minimum path cover problem is reformulated into a *weighted* minimum path cover problem. That is to find a path cover $P = \{p_1, p_2, ..., p_n\}$ with the objective function

$$min \sum_{p \in P} \phi(p) \tag{2}$$

In terms of the NRS detection, a solution to this combinatorial optimization problem corresponds to a set of NRS merged from the contig sets of many individuals.

4.2.2 A greedy heuristic

PopIns2 introduces a greedy heuristic as a practical solution to solve the weighted minimum path cover problem on a CDBG built from a set $\mathcal{S} = S_1, S_2, ..., S_n$ of contig sets. Each contig set $S_i \in \mathcal{S}$ is assembled from the unaligned and poorly aligned reads of an individual's genome and corresponds to a color in the CDBG. The key idea is to initiate a depth-first search (DFS) from *sources* of the graph, i.e. vertices that have at least one successor but no predecessors. During the traversal of the DFS, the decisions which vertex to proceed with at branching nodes is prioritized by the Jaccard index of color bitvectors. The traversal continues until a *sink* is found, i.e. a vertex that has at least one predecessor but no successors, or aborts in case a local substructure of the CDBG cannot be resolved by the traversal. In case the traversal finds a sink, the path is checked whether its vertices contain a minimum number of previously undiscovered k-mers. If the path passes the test its vertices are concatenated and one final genomic sequence is returned. A pseudo code of the DFS' main routine is included in the appendix (Algorithm 1).

More formally, let G=(V,E,C) be a compacted CDBG of input set $\mathcal{S}$ of contig sets and k-mer size k. Every vertex $u \in V$ has a traversal state that can be either *seen* or *unseen*. Initially, all vertices are in the state of being unseen. Further, a unitig set $\mathcal{D}$ records which k-mers have been covered by a path. Note that every vertex has at most four predecessors and successors as the CDBG is built with genomic sequences (chapter 2.2).

DFS initialization. In the initialization step of the DFS all nodes $u \in V$ are checked whether they have predecessors and successors. This check has three potential outcomes:

1. Vertex u has no predecessors and no successors, i.e. u is a *singleton*
2. Vertex u has predecessors
3. Vertex u has only successors, i.e. u is a source

If u is a singleton then the genomic sequence of u is returned as a final genomic sequence without further ado. Since a singleton, by its definition, is isolated from other vertices in V none of its k-mers can be members of the path cover already. If u has predecessors the vertex is ignored in the initialization step. If u has only successors then u is a source and is passed to the recursion step of the DFS.

DFS recursion. Let u_c be the currently visited vertex of the traversal. At first, vertex u_c is checked whether it has any unseen successors $\mathcal{N} \subset V$. If $\mathcal{N} = \varnothing$ then u_c is a sink. In that case the number of k-mers in the path p from source to u_c that are not members of $\mathcal{D}$ is determined. If that number of novel k-mers in p exceeds a user-defined threshold τ then the novel k-mers are added to $\mathcal{D}$ and the NRS $\omega(p)$ (see chapter 2.4) is returned.

If u_c has a non-empty set of unseen successors $\mathcal{N}$ the traversal algorithm has to make a decision how to continue the traversal through the CDBG. This decision is a crucial design choice of the algorithm since it determines the order of the vertices in the path and, consequentially, the genomic sequence of the NRS. The decision which successor to continue with is made by utilizing the colors of the CDBG. For each successor $n \in \mathcal{N}$ of u_c an edge weight is computed using the color vectors $last(u_c)$ and $first(v)$ corresponding to the last k-mer of vertex u_c and the first k-mer of vertex v, respectively. The edge weight is defined as $1 - J(last(u_c), first(v))$ where the function J is the Jaccard index defined over bitvectors. Then, the recursive traversal at u_c is continued with the successor returned from

$$\arg\min_{v \in \mathcal{N}} \{1 - J(\text{last}(u_c), \text{last}(v))\} \tag{3}$$

If all successors of the current vertex have been seen before the recursion takes one step back and continues with the next best successor returned from Equation (3). The recursion of the DFS continues until a sink is found. In that case the traversal states of all vertices in G are reset to unseen and the DFS algorithm is continued with the next source. The final output of the greedy heuristic is a set of sequences (supercontigs) which correspond to a set of maximal unitig paths.

4.3 Implementation of PopIns2

This chapter walks the reader through the practical implementation of the methods described in chapter 4.2, explains optional features and provides hints and tips for large-scale data processing with PopIns2. The majority of the code refactoring in PopIns2 was done

in the merge module. Thus, this chapter will predominantly focus on this module.

To follow the subsequent explanations of this chapter the reader requires a fundamental knowledge of high-level programming languages and common programming paradigms. Fundamental knowledge about the Unified Modeling Language (UML) and high performance computing environments is beneficial but not strictly necessary. For an in depth understanding of some selected design pattern, knowing a few terms and concepts that are specific to the C++ programming language is inevitable. As this chapter does not aim to educate about technical details of the C++ programming language, references are provided at the given place.

4.3.1 Design pattern

The new merge module of PopIns2 is designed with the Object Oriented Programming (OOP) paradigm. All the functionality to generate paths from the CDBG, and subsequently the NRS, is implemented in classes and their member functions (Figure 18). During the merging step there are five classes whose instances interact with each other.

The class with the most and most complex code is the ExtendedCCDBG, which stands for *Extended Colored and Compacted de Bruijn Graph*. Later, the paragraph about the UnitigExtension class is going to explain why the implementation of the ordinary CDBG needed to be extended even further. The essential functionality of the ExtendedCCDBG is the DFS traversal (chapter 4.2.2) and prioritizing the successors of a unitig using the Jaccard index of colors bitvectors. Once the graph is built the only member function that needs to be called for the invocation of the DFS is *traverse*. The traverse member function is a wrapper function for the DFS initialization phase. It iterates over every vertex in the CDBG and if a vertex $u \in V$ is a source then the DFS is initiated from u. All the other functionality for the traversal, like determining the Jaccard index between two color bitvectors, ranking the neighbors or the DFS recursion, is hidden in private member functions that are called from within the traverse function.

The ExtendedCCDBG class itself does not explicitly implement any functionality to inspect the genomic sequence of a vertex, the predecessors and successors of a vertex or the colors of a k-mer. Instead, the ExtendedCCDBG inherits functionality from the *ColoredCDBG* class (Figure 18). ColoredCDBG stands for *Colored and Compacted de Bruijn Graph* and is a template class from the Bifrost API (chapter 2.6). The ColoredCDBG again inherits functionality from the template class *CompactedDBG*. Together the underlying ColoredCDBG and CompactedDBG classes provide all the functionality for the ExtendedCCDBG that is required to inspect the unitig sequence of a vertex, find a unitig via a given k-mer in constant time (see data structures in chapter 2.6), determine the neighbors of a vertex and inspect the color bits of every k-mer in a unitig. Further, the ExtendedCCDBG inherits the basic functionality for reading and writing FASTA and FASTQ files for genomic sequences, the Graphical Fragment Assembly (GFA) format for DBGs, the Bifrost-specific BFG_COLORS format for a compressed color matrix,

constructing a DBG for a given k-mer and minimizer length, simplifying a DBG (that is deleting tips and singletons shorter than a given minimum number of k-mers) and for annotating k-mers with the colors of their corresponding input samples.

In the Bifrost API, the unitigs of the ColoredCDBG are instances of the class *UnitigColorMap* (abbreviated with *ucm* in function parameters in Figure 18). Although the UnitigColorMap class complies with all the aforementioned functionality of the ColoredCDBG it had to be extended in order to use it for the DFS traversal in the PopIns2 merge algorithm. In PopIns2, the UnitigColorMap of the ColoredCDBG (and subsequently of the ExtendedCCDBG) needs to be extended by two properties. The first property is a unique identifier (ID). The ID of a unitig is implemented rather for convenience and traceability since any k-mer of a unitig could be used as a unique identifier too. The second and strictly necessary additional property of the unitigs is the DFS states, i.e. a member variable of the UnitigColorMap that indicates whether the vertex has been visited before. Note that in PopIns2 each vertex has two DFS states because the CDBG is bidirectional and forward and reverse complement sequence require individual DFS states.

In many higher programming languages a straightforward way to associate additional data to given (unique) elements is to use a map (e.g. in C/C++/Java) or dictionary (e.g. in Python). This way a map can store any single of a unitig's k-mers as a key and store its corresponding ID or DFS states as values. However, for this use case the Bifrost API has a sophisticated solution to circumvent the usage of additional data structures. The CompactedDBG and ColoredCDBG are class templates[16] whose instances can be constructed with a template parameter[17]. The CompactedDBG implements a mechanism that associates an instance of the template parameter type to every vertex (UnitigColorMap) of the graph. From a practical point of view, this means that the CompactedDBG and ColoredCDBG can be constructed with a data type that extends the UnitigColorMap class with auxiliary data. In contrast to additional data structures like a map, now the auxiliary data resides within the graph. One important constraint in the Bifrost API is that the template parameter type of the CompactedDBG or ColoredCDBG is a child class of the type *CCDBG_data_t* (Figure 18). The ExtendedCCDBG introduced in the beginning of this chapter is a child class of the ColoredCDBG with a template parameter of type *UnitigExtension*. An instance of the UnitigExtension class contains a unique unitig ID and two DFS states for every vertex in the graph.

A class that has a frequent interaction with the ExtendedCCDBG is the *Traceback* class. An instance of the Traceback class records the vertices that have been visited from the source to the current node. Upon request the Traceback class concatenates the unitig sequences of the vertices whilst taking into account the $k-1$ overlap and writes the final genomic sequence (supercontig) to a file.

[16] https://en.cppreference.com/w/cpp/language/class_template
[17] https://en.cppreference.com/w/cpp/language/template_parameters

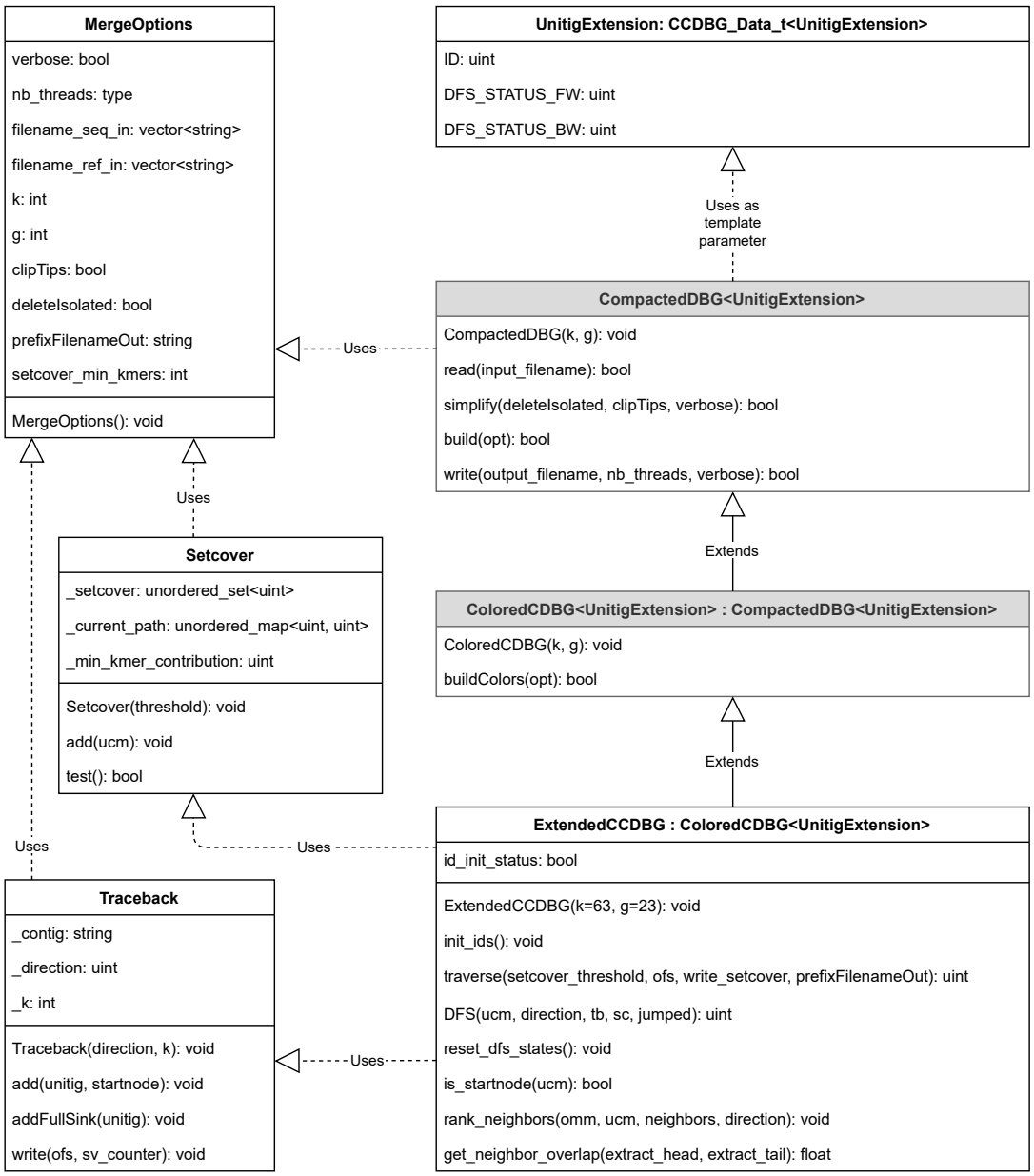

Figure 18: Class diagram of the PopIns2 merge module. Every box in the diagram denotes a class in the program. Classes with a grey background in the header are implemented by the Bifrost API. If a horizontal line splits the class description, entries above the line denote member variables and entries below the line denote member functions (also written with parentheses before the return type). The *Extends*-relation arrow denotes a class inheritance from derived class to base class. The *Uses*-relation arrow denotes that a class interacts with another class in one or multiple of its member functions.

The second class that interacts during the DFS traversal is the *Setcover* class. Its purpose is to regulate the redundancy in the set of supercontigs. The *test* member function of the Setcover class checks whether the unitig sequences of a given path contribute a minimum number of novel k-mers to the set of supercontigs till this point of the traversal. The Setcover class is a practical solution of PopIns2 for property (3) in chapter 4.1.2.

For completeness, Figure 18 also includes the MergeOptions class. The MergeOption class is only a data class containing the command line parameters. All three major classes involved in the traversal (ExtendedCCDBG, Setcover and Traceback) obtain their parameters and thresholds from this data class, predominantly at construction time.

4.3.2 Control flow

In the new merge module of PopIns2 there are primarily three classes whose instance interact with their member functions. Figure 19 shows a flowchart about the chronology of events within the merge module and how the classes ExtendCCDBG, Setcover and Traceback interact. The focus of the subsequent description are the mechanisms of the DFS traversal. Details of the implementation to wield the bidirectionality of the graph are omitted.

Graph build. Initially, an instance of the ExtendedCCDBG reads one or many files of contigs in FASTA or FASTQ format, e.g. from the PopIns2 assembly module, and builds a colored and compacted de Bruijn Graph $G = (V, E, C)$ from the contigs using a given k-mer size k. Building the graph G is composed of three subroutines. First, a compacted DBG $G' = (V, E)$ is built from the contigs. Second, G' is simplified, i.e. all tips and singletons are deleted from G' (chapter 2.6). Third, a color matrix C (chapter 2.4) is generated for G', where each file of contigs is an input set (color). Together, G' and C define G. Once G is built, simplified and annotated with colors, every unitig receives an unique identifier (ID).

Graph traversal. The DFS traversal through G is implemented in the *traverse* function of the ExtendedCCDBG. The function first constructs an instance S of the Setcover class and starts a program loop over all unitigs in G (red rhombus in Figure 19). Let u be the control variable, i.e. the currently observed unitig of the loop. The function *is_startnode* checks if u is a source. If this check evaluated to false then the loop continues with the next unitig. If this check evaluates to true then an instance T of the Traceback class is constructed and the *DFS* function is called with u as initial vertex. Next, the function *hasSuccessors* checks whether u has successors $\mathcal{N}$. If this check evaluates to false then u is a sink too and, consequentially, a singleton. In that case T reports the entire genomic sequence of u as supercontig and the ID of u is added to S. The latter is not strictly necessary for the traversal since a singleton cannot be observed again but the PopIns2 merge module can optionally report the entirety of visited unitigs from S for a manual inspection of the CDBG. Until here, all three cases of the DFS initialization in chapter 4.2.2 are detected.

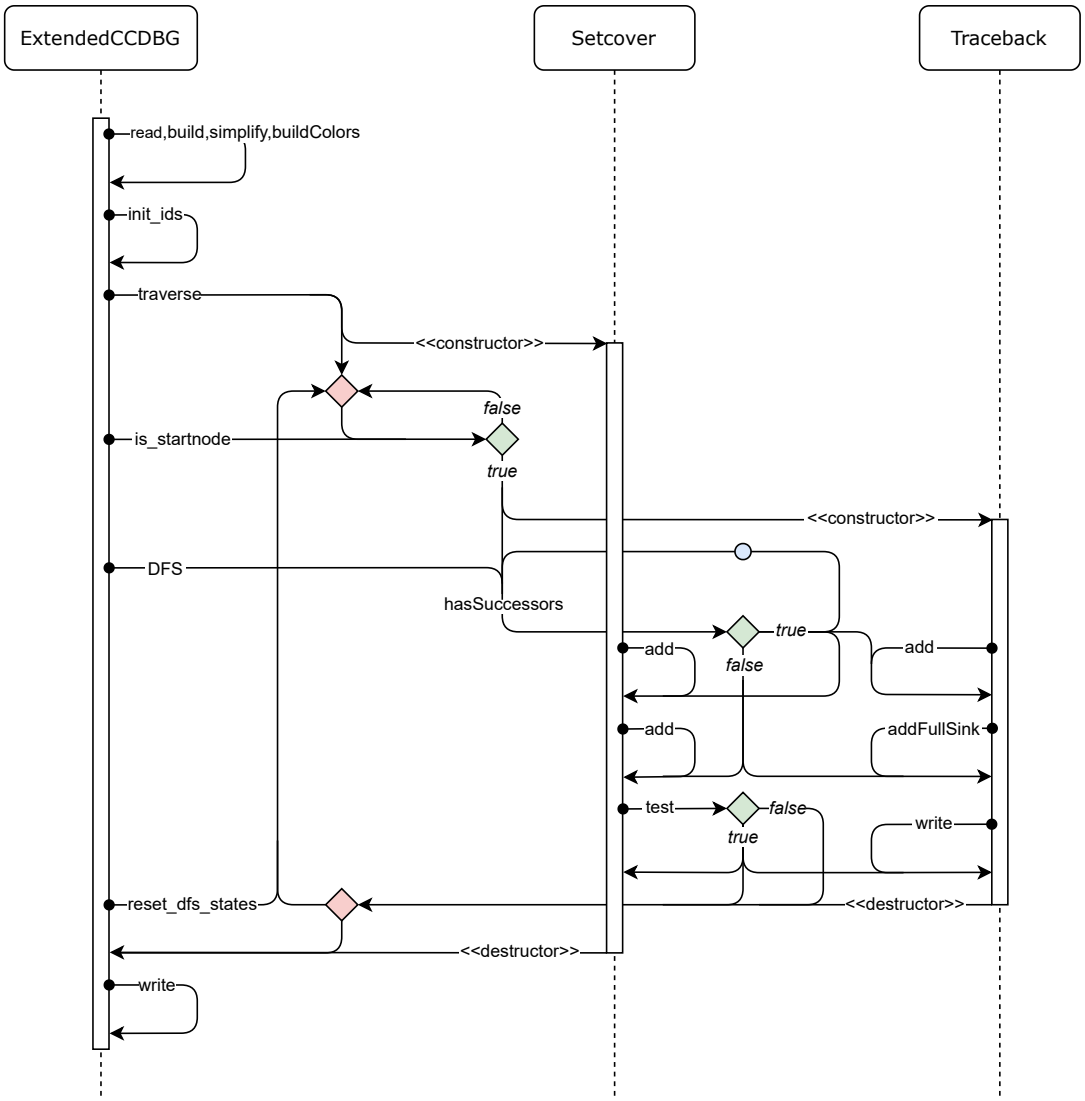

Figure 19: Flowchart of the PopIns2 merge module. Each solid long bar in the flowchart denotes the lifetime of a class instance from top to bottom. The black dots at the class instances denote an invocation of a member function. The red tiles denote the start and end point of a for-loop over all unitigs of the ExtendedCCDBG instance. The green tiles denote that the flow can follow two different paths at this point depending on the incoming boolean return statement of the preceding function. The blue dot denotes a recursive invocation of the DFS function.

If the check for the presence of neighbors evaluates to true then Equation (3) is evaluated to find the unseen neighbor $n_c \in \mathcal{N}$ for the recursive continuation of the DFS. In Figure 19 the blue dot symbolizes and hides the details of the steps associated with the DFS recursion, e.g. marking u as seen and passing n_c to the next call of DFS. Also, every time an eligible successor is determined the current unitig has its genomic sequence added to T and its ID added to S. The recursion goes on until a sink is found or local substructures in the graph cannot be resolved.

If the DFS reaches a sink, S tests whether the unitigs in the path p from source to sink exceeds a threshold τ of novel k-mers with respect to the already reported supercontigs. Here, the *test* function sums up the number of k-mers of all unitigs in p whose IDs are not in S yet. If the test evaluates to true, the genomic sequence $\omega(p)$ is reported as supercontig, the unitig IDs of all unitigs in p are permanently stored in S and T is deleted. Only the latter happens if the minimum number of novel k-mers in p does not exceed τ.

This point marks the end of instructions related to the current control variable u of the loop. If the iteration of the loop is not finished then all DFS states are reset to unseen and the iteration starts with the next u. If the iteration of the loop is finished, S is deleted and the CDBG can optionally be written to disc.

4.3.3 Bridging unitigs of low entropy genomic sequence

This subsection examines a feature of the PopIns2 software that masks subsequences of low complexity (chapter 2.2) in NRS, i.e. instead of reporting the actual genomic subsequences of the NRS that are of low entropy, they are replaced with generic characters. This feature was successfully tested for its principal functionality and applicability on real genomic sequence data. However, the extend of genomic sequence being masked with different entropy thresholds was not thoroughly evaluated in [Krannich *et al.*, 2021] and the decision whether low entropy masking is desired is left to the user. By default no low entropy masking is applied by PopIns2.

In some genomes, including the human genome, sequences of low entropy (e.g. short tandem repeats, homopolymers or poly-A tails) are observed frequently [Dechering *et al.*, 1998; Willems *et al.*, 2014]. These sequences contain a notably less diverse spectrum of nucleotide dimers. Consequentially, sampling k-mers from low entropy sequences results in many highly similar k-mers. If these similar k-mer are inserted into a DBG they form characteristic topologies (Figure 20a,b). Those characteristic topologies tend to further cluster into larger and densely interconnected components (Figure 20c), here termed *low entropy connected components* (LECC). The DFS procedure described in chapter 4.2 can make poor traversal decisions in subgraphs of highly similar and abundant k-mers from LECCs and hence return incorrect supercontigs, i.e. the reconstructed sequence is erroneous with respect to the true biological sequence. The same consequence of low entropy sequence from real biological data was already discovered in the original version of

PopIns, even though the algorithmic approach was different. The approach of PopIns to handle sequences of low complexity is to exclude contigs below a certain entropy threshold from the merge procedure. The masking option of PopIns2 intends to not ignore the entire sequence but to mask its low entropy k-mers during the creation of the supercontigs.

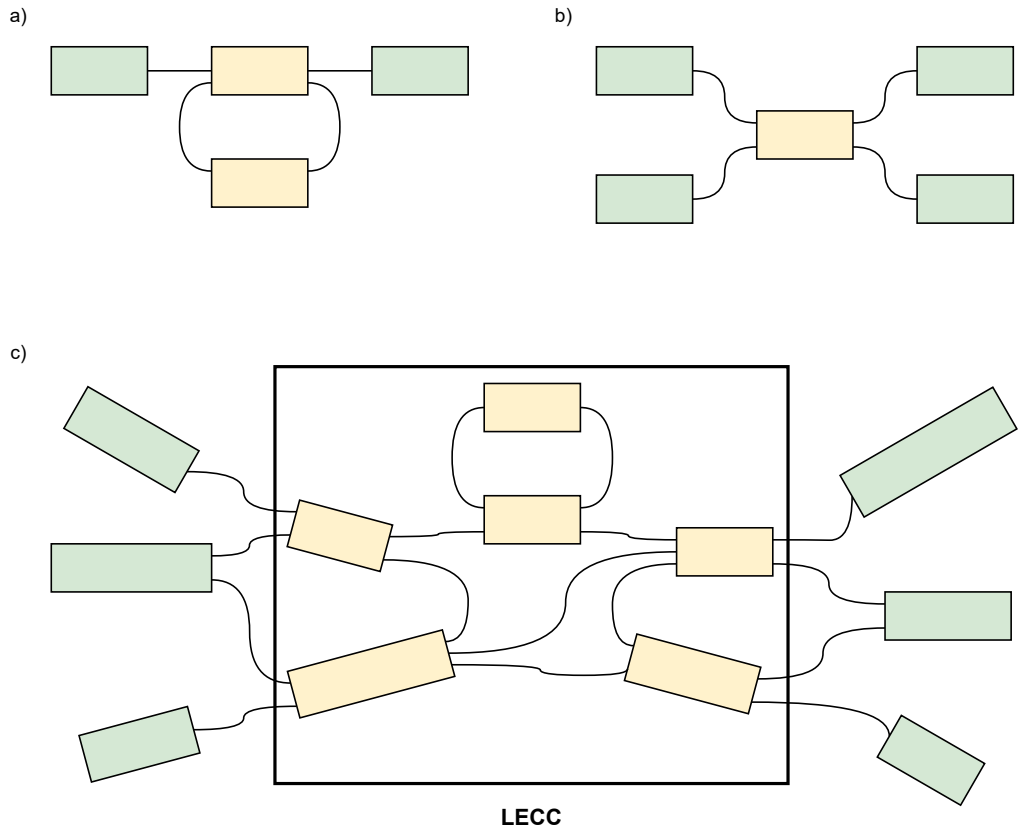

Figure 20: Frequently recurring DBG topologies of low entropy genomic sequences. Vertices in yellow are unitigs with a sequence entropy below a given threshold. Figure a) and b) show characteristic topologies in a DBG that are caused by unitigs of low entropy. Figure c) shows a densely connected subgraph of unitigs with a low entropy sequence, called *Low Entropy Connected Component* (LECC).

Finding partners around LECCs. If the low entropy masking is activated by providing an entropy threshold other than zero, all unitigs whose sequence entropy is below the given threshold (from here called *low entropy unitigs*) are annotated. Next, all LECCs are identified. Finding the LECC of a given low entropy unitig in a DBG is trivially realized by starting a depth-first search. Initially, any vertex u of the DBG is marked as visited. If u is a low entropy unitig, it is assigned a unique LECC identifier. All

neighbors of a low entropy unitig u (at most eight) that are low entropy unitigs as well and have not been visited yet are annotated with the same LECC identifier as u and marked as visited in the DFS. The latter step is recursively repeated with the neighbors of u until no more neighbors can be visited. If after the recursion another vertex of the DBG is still not visited, it becomes the new u to start the DFS from. If all vertices of the DBG are visited then every low entropy unitig is assigned a unique LECC identifier.

For each LECC L all unitigs $\mathcal{U}_L$ which border L (so each $u \in \mathcal{U}_L$ is not a low entropy unitig) are stored together with the k-mer k_L that connects to L. For every $u \in \mathcal{U}_L$ another DFS is initiated through L to find all accessible unitigs $\mathcal{P} \subseteq \mathcal{U}_L$, called *Partners*. *Accessible* here means there exists a path from source u to sink $p \in \mathcal{P}$ through L. Next, the Jaccard index of the color bitvectors of $k_L \in u$ and each k_L of $p \in \mathcal{P}$ is computed. As a result, a map $\mathcal{M}_L$ stores the associations of every u with its partner p that has the highest color match.

DFS traversal with jumps. Once the $\mathcal{M}_L$ is computed for every LECC in the graph the PopIns2 merge module starts the main DFS traversal to generate the set of supercontigs. Let u be the currently observed vertex of the latest recursive step of the DFS. With low entropy masking enabled, if the unseen successor v of u returned from Equation (3) is annotated as low entropy sequence then v is a vertex of a LECC L and u is a vertex that borders L. Hence, u can be looked up in the map $\mathcal{M}_L$ together with its partner p of the highest color match. Instead of continuing with v, the DFS continues with p (the DFS is said to "jump" over the LECC). Later, when function ω concatenates the unitig sequences of the path, a jump is encoded as 'N' characters of length k.

Masked sequences. The decision to mask low entropy sequence is left to the user. By default the masking process is disabled. With the command line parameter $-e$ of the PopIns2 merge subcommand the user can specify an entropy threshold for the masking and jumping process. Further, PopIns2 offers a command line parameter $-l$ to write out a CSV file with all vertex IDs of the CDBG whose genomic sequence were annotated as low entropy. That CSV file, together with the GFA file of the CDBG, can be loaded into *Bandage* [Wick *et al.*, 2015] to visually inspect the masking process.

4.3.4 A multi-k construction algorithm for CDBG

This chapter examines a prototype method that builds the CDBG directly from reads instead of contigs and therefore circumvents the contig assembly per individual. The method is implemented as part of the PopIns2 software and was successfully tested for its principal functionality. However, tests with simulated data at the early stage of its development did not show improvement on benchmarks at intermediate checkpoints of the PopIns2 workflow (chapter 5.1.3). Therefore, the implemented feature remains experimental and is not thoroughly documented or published. Also, note that the current implementation was not yet optimized for performance but for traceability.

In the standard workflow of PopIns2, one of the first steps is a contig assembly per individual of all its reads which are unaligned and poorly aligned to the reference genome (chapter 2.7). The resulting sets of contigs are then used as input for the merge module. This implies that the contigs which go into the merge module, and hence into the CDBG, were already subject to certain parameter settings and the algorithmic design of the external assembly software that is used for the contig assembly. All the external assembly software that was assessed during the development of PopIns2 is tailored to whole-genome assembly and therefore optimized towards continuity (see Evaluation statistics in 2.5). In contrast to whole-genome assembly which aims to reconstruct only few but long chromosomes, the contig assembly of PopIns2 aims to reconstruct many but rather short genomic sequences, the NRS.

Additionally, each external assembly software has its dedicated methods to reduce the complexity of the assembly problem and to eliminate technical artifacts in the sequence data. Common methods for this are filters for k-mer abundance, trimming of spurious vertices from graphs or exclusion of sequences below a minimum length. Not always are all of these methods known to the user or can be trivially deactivated.

To circumvent that the aforementioned properties of external assembly software affect the merge procedure of PopIns2, an alternative method was developed to build the CDBG from raw genomic reads, called the *multi-k method*. The multi-k method uses the unmapped and poorly mapped reads instead of the contigs per individual. Its general idea is to build the CDBG from several k-mer lengths k to combine the graph connectivity of small k-mer sizes with the unitig continuity of large k-mer sizes. In the PopIns2 workflow (Figure 14), the multi-k method can be used between the assemble and merge module as a substitution for an external assembly software (which is part of the assemble module). To save computational resources, the assemble module can disable the usage of an external assembly software via the *--skip-assembly* flag. Mind that the usage of the assemble module per individual is still required as it returns the FASTQ files with unaligned and poorly aligned reads. The result of the multi-k method is a CDBG that can be loaded into the merge module.

The iteration. The multi-k method is an iterative procedure (Figure 21) that requires an initial k-mer size k, a maximal k-mer size k_{max}, a k-mer step size Δ_k and a set I_{in} of FASTQ or FASTA files (one file per individual). The first step is to build a CDBG G from I_{in} and k. In the implementation of the multi-k method this is done with the Bifrost API (chapter 2.6). After G is built, graph simplifications can be applied. Next, k is increased by Δ_k. If k is now larger than k_{max} the iteration ends at this point. Otherwise a function f decides for each unitig if it is selected for the next iteration. A map stores for each color (individual) of G which unitigs are selected for the next iteration. The selected unitigs are written into a temporary file per color. Let I_k be the set of temporary files for the current value of k. The input data I_{in} is updated by concatenating the sequence file

of each individual with its corresponding unitigs from I_k. After that the iteration starts the next cycle.

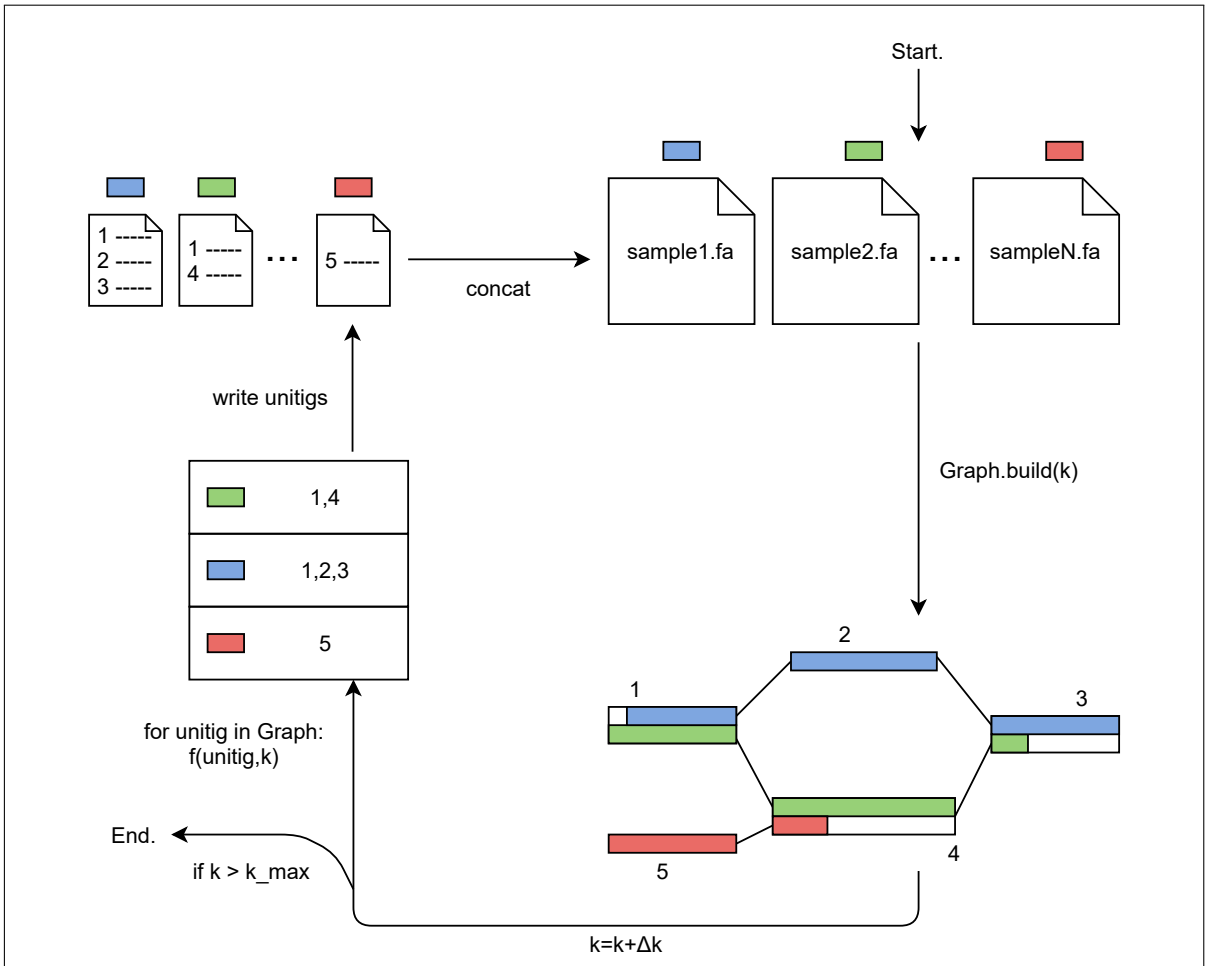

Figure 21: Sketch of the iterative multi-k approach to build a CDBG. The three core mechanisms of the iterative process are building a CDBG, increasing k and choosing unitigs to be taken into the next iteration.

Choosing unitigs from a CDBG. The iterative procedure is not a novel concept for DBG. In fact, it has been implemented in multiple programs for DBG based assembly [Chikhi and Rizk, 2013; Li *et al.*, 2015] or variant calling [Turner *et al.*, 2018]. In PopIns2 it is attempted to transfer this concept to a CDBG. The alert reader noticed that in the last paragraph the decision criterion how to select the unitigs for the next iteration was hidden in the function f. This function is trivial for plain DBGs, where $f : unitig \times k \to \{0, 1\}$ simply decides whether the length (in base pairs) of the unitig is larger or equal to k.

However, in a CDBG using this trivial function is likely to introduce a lot of additional genomic sequence to the next iteration that actually was not present in the graph. This effect can be illustrated with the example in Figure 21. In the CDBG the unitig with the identifier 3 (UID3) has two colors, i.e. k-mers of UID3 originate from two individuals. All k-mers of UID3 are present in the individual corresponding to the blue color but only few k-mers of UID3 are present in the individual corresponding to the green color. Let i be the counter of the iterations of the multi-k method. If function $f(UID3, k)$ is evaluated and returns true for the current value of k then UID3 in its entirety is utilized for the graph construction in iteration $i+1$. Therefore, the questions remains which individuals should add UID3 to their input data for iteration $i+1$. If UID3 is naively added to the set of sequences of each individual that contributed a color to UID3 in iteration i then, in the example above, the individual corresponding to the green color will gain k-mers in iteration $i+1$ that are actually not its original input data.

The challenge in finding a function f for a CDBG is to incorporate the color information into the decision which individuals that contributed k-mers to a particular unitig in iteration i will receive its full or partial unitig sequence in iteration $i+1$.

The idea for function f in the prototype of the PopIns2 multi-k method is a coefficient $\frac{1}{\delta}$ that defines what minimum fraction of k-mers in a unitig have to be present in an individual in order to add the entire unitig to the individual's input data for the next iteration. However, using a minimum fraction of a unitig as threshold means that f has to check at least $\lceil |unitig| \cdot \frac{1}{\delta} \rceil$ bits in the color matrix C per individual for all unitigs. This number of bits to check rapidly becomes impractical for a large number of individuals. To reduce the number of bits that f has to check per unitig, a *k-mer sampling* scheme was implemented in PopIns2. Let $\delta \in \mathbb{N}_{\geq 2}$ and $g : \delta \times l \rightarrow P$ a function that returns an ordered set P of k-mer positions from a unitig of k-mer length l. Function g returns a set of $\delta + 1$ equidistant k-mer positions between and including 0 and $l-1$. For instance, g returns the set $\{0, \lceil \frac{l-1}{2} \rceil, l-1\}$ if $\delta = 2$. It follows that an interval $[p_i, p_{i+1}]$ of any two contiguous numbers $p_i, p_{i+1} \in P$ span a range of $|unitig|/\delta$ k-mer positions in a unitig.

Now, f uses the sampled k-mer positions P to estimate whether there are enough k-mers of a certain color in the unitig in order to satisfy the minimum threshold $\lceil |unitig| \cdot \frac{1}{\delta} \rceil$ and add the unitig sequence to its input of iteration $i+1$. Under the assumption that k-mers with a certain color appear continuously in the unitig, f can test the minimum threshold by only checking $\mathcal{O}(\delta)$ bits per color. This is done by testing whether there exists a pair (p_i, p_{i+1}) of any two positions $p_i, p_{i+1} \in P$ (p_i, p_{i+1} are contiguous in P) where their corresponding k-mers in the unitig both have the color bit set for a particular individual.

The prototype of the multi-k method in PopIns2 was implemented with $\delta = 2$, i.e. an estimated 50% of the k-mers of a unitig have to be annotated with a particular color in order for the unitig to be added to the sequences of the corresponding individual. Figure 22 illustrates f with $\delta = 2$ applied to a unitig of length $l = 11$ and three colors.

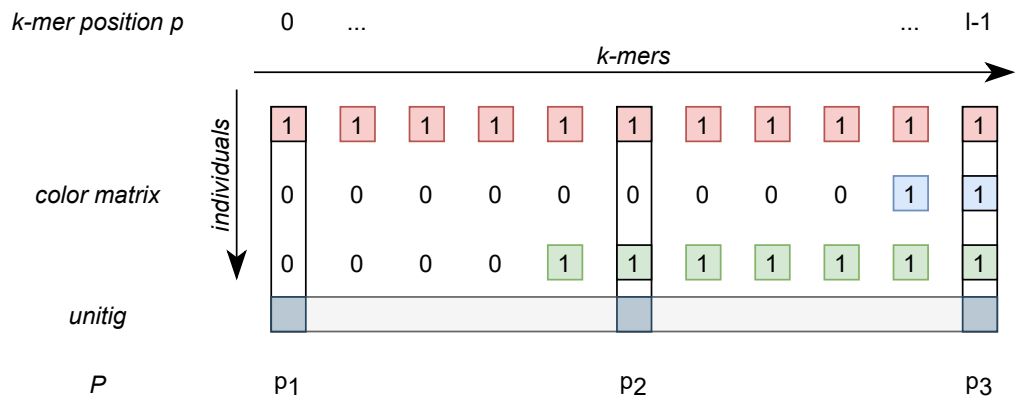

Figure 22: Sampling and inspection of k-mer positions in a unitig. The grey bar is a unitig of k-mer size $l = 11$. Each k-mer has a color vector of three bits (red, blue, green). The sampled unitig positions P are $p_1 = 0$, $p_2 = 5$ and $p_3 = 10$.

It follows that there are $|P| = \delta + 1 = 3$ sampled k-mer positions ($P = \{0, 5, 10\}$) and $\lceil |unitig| \cdot \frac{1}{\delta} \rceil = \lceil 11 \cdot 0.5 \rceil = 6$ is the minimum threshold. Unitig positions contiguous in P are the pairs $(0, 5)$ and $(5, 10)$. Therefore, if the bits for a particular color are set for the k-mers at positions 0 and 5 or at positions 5 and 10 then the unitig is added to the sequences of the corresponding individual. In Figure 22 the individual corresponding to the red color has the bits set at the sampled positions 0, 5 and 10 such that the unitig will be added, the individual corresponding to the blue color has the bit set only at the sampled position 10 such that the unitig will not be added and the individual corresponding to the green color has the bits set at the sampled positions 5 and 10 such that the unitig will be added.

To highlight the benefit of the k-mer sampling scheme in terms of checked bits, let's assume that f would check every k-mer position in a unitig from 0 to $l - 1$ for every color without the k-mer sampling. In the given example of Figure 22 function f without k-mer sampling needs to find a continuous run of at least six k-mers in order to exceed the minimum threshold. Hence, f would check six red bits (results in true), six blue bits (results in false) and ten green bits (results in true). In contrast, f with the k-mer sampling scheme g checked the two red bits at p_1, p_2, the three blue bits at p_1, p_2, p_3 and the three green bits at p_1, p_2, p_3.

4.3.5 The Alignment score factor

This subsection describes a minor update in the implementation between the first and second major release of the PopIns workflow. This minor update gives the freedom to the user to adjust a crucial selection criterion for detecting unmapped reads in an individual

before its contig assembly. In contrast to the previous chapters, this change affects the assemble module of PopIns2.

The assemble module implements a set of criteria [Kehr *et al.*, 2016, 2017] to evaluate whether a read has a low quality alignment to a given reference genome and consequentially is added to the set of reads that is considered for the contig assembly. One important criterion of that set includes the *alignment score* of a read. As the PopIns2 workflow starts from a set $\mathcal{R}$ of reference aligned short-reads of every individual (chapter 2.7) it is expected that every $r \in \mathcal{R}$ has a corresponding alignment score AS_r. Assuming that r is part of an unobtrusive read pair (in PopIns defined as a pair of reads of oriented towards each other and with less than 1000 base pairs inner distance) then the alignment quality of r is considered low if the inequation

$$AS_r < ASF \cdot |r|$$

holds. The ASF is the *alignment score factor*, where ASF$\in [0, 1]$, that can be specified via the command line interface of PopIns2 (*--alignment-score-factor FLOAT*). With a growing ASF the reads with a high AS are still considered having a low alignment quality. In other words, a higher ASF will filter more reads into the set of unaligned and poorly aligned reads per individual.

4.3.6 Availability and resources

This last chapter of the methods summarizes all available resources to get started with PopIns2 and provides practical knowledge to apply PopIns2 to large data sets.

Source code, dependencies and test data. PopIns2 was built upon the source code of its predecessor PopIns [Kehr *et al.*, 2016]. The new software, PopIns2, is available from a separate Github project[18]. PopIns2 is written entirely in the C++ programming language and was developed for the usage on 64-bit UNIX based operating systems.

The source code can trivially be compiled via a provided Makefile once all dependencies are installed and specified accordingly in a configuration file. As dependencies, PopIns2 requires the Bifrost [Holley and Melsted, 2020] and seqAn [Reinert *et al.*, 2017] C++ libraries, the BWA [Li, 2013] read alignment software, the Sickle read trimming software[19] and at least one of the three supported genome assemblers (Minia [Chikhi and Rizk, 2013], Velvet [Zerbino and Birney, 2008] or SPAdes [Bankevich *et al.*, 2012]). Minia is automatically downloaded and supported with the recommended installation process of PopIns2. Mind that Minia requires a Python2.7 language interpreter. Further details about the installation and application of PopIns2 can be found at the Github project page.

[18] https://github.com/kehrlab/PopIns2
[19] https://github.com/najoshi/sickle

The Github project also refers to test data[20] to apply PopIns2 to a minimum working example (MWE). The MWE comprises simulated reads from three human individuals and a modified genomic sequence of the human chromosome 21. If PopIns2 is applied to the data of the MWE with default parameters, the user will obtain a small set of NRS genotyped for all three individuals.

Workflow language support. PopIns2 is implemented as a program of multiple, consecutively executable modules (Figure 14) which seamlessly integrate into workflow languages like Snakemake [Mölder *et al.*, 2021]. The Github project of PopIns2 refers to another auxiliary repository[21] that contains a generic Snakemake workflow for a quickstart of PopIns2 with minimal dependencies. That workflow only requires a reference genome, a list of sample identifiers and a predefined project structure. Figure 23 shows the Snakemake workflow for the three individuals of the MWE. The workflow has a directed non-cyclic progression of the PopIns2 modules where every module reports its successful or unsuccessful execution status to its successor and the final control instance, called *all*-rule.

SIMD and hardware optimization. All building blocks of PopIns2 were carefully evaluated and chosen for its computational efficiency, with no exception to new merge module and the external Bifrost library. The graph data structures implemented in Bifrost are highly optimized for their performance in terms of build and accession time. Among those optimizations is the usage of *single instruction multiple data* (SIMD operations, chapter 2.6) instructions which are, in practice, specific to the CPU architecture.

As PopIns2 is designed to process large data sets, it is recommended to distribute its workflow among a high performance computing (HPC) environment, like a cluster environment of many computational units (nodes). However, with the SIMD instructions enabled (default behaviour) the executable, binary program of PopIns2 is specific to the CPU architecture of the machine it was compiled on. Consequentially, running PopIns2 on a HPC cluster environment of heterogeneous nodes likely leads to a failure at some point of the workflow. It is left to the user to distribute the PopIns2 workflow among a suitable HPC cluster environment. Optionally, with noticeable loss in computational performance, SIMD instructions can be disabled in Bifrost and PopIns2.

[20] https://doi.org/10.5281/zenodo.4890793
[21] https://github.com/Krannich479/PopIns2_snakeproject

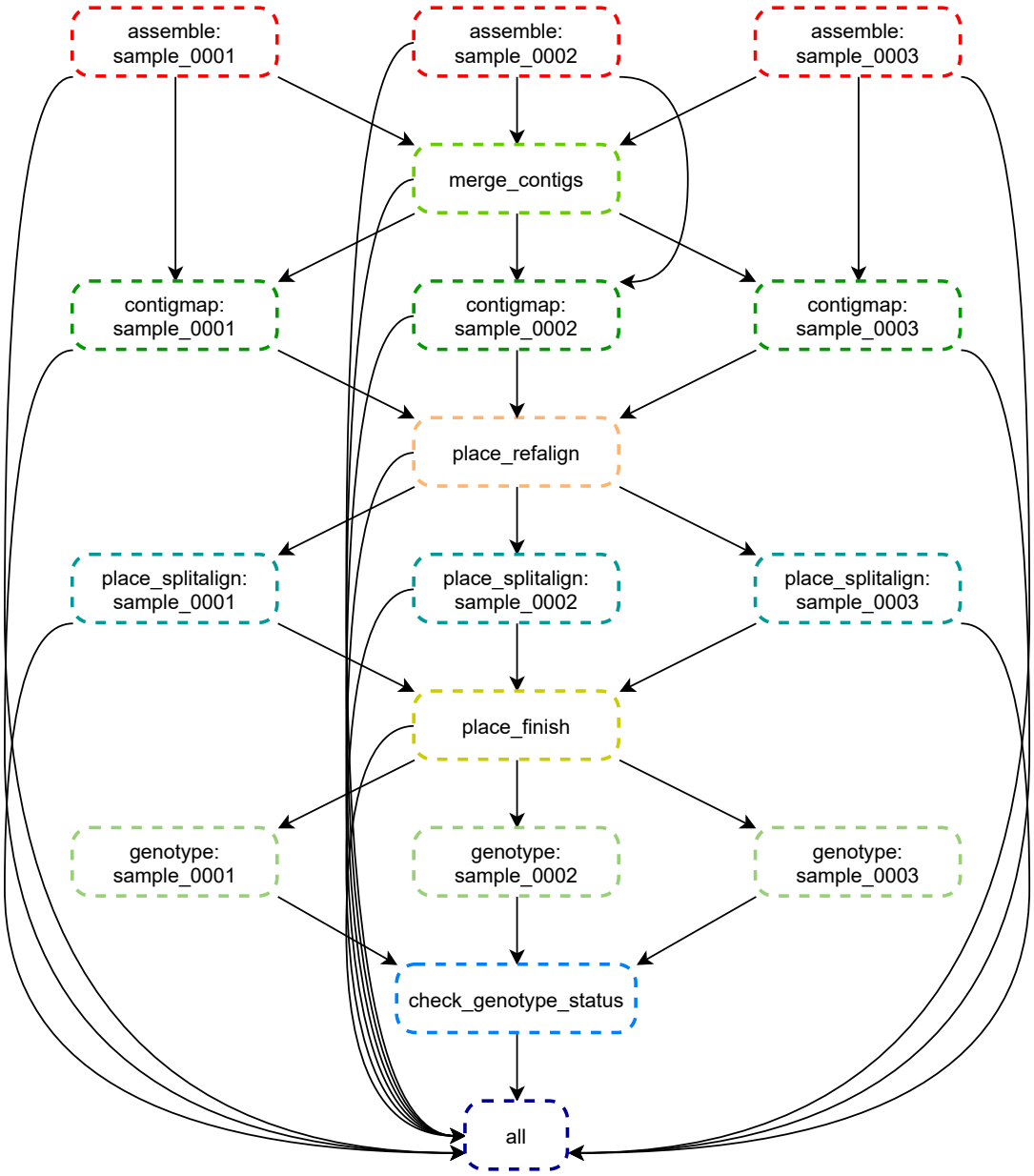

Figure 23: Snakemake workflow of the PopIns2 modules applied to three individuals.

5 Results

This fifth chapter describes the evaluation process of PopIns2 and assesses results from different intermediate stages of the program as well as from its workflow in total. First, PopIns2 is assessed using simulated genomic sequences of many individuals such that the NRS detection can be evaluated on a know variant callset. Thereafter, PopIns2 is applied to different cohorts comprising real genomic data where particularly the aspect of computational performance is highlighted but also the insertions after the entire workflow are assessed. To rate the results and performance of PopIns2, it is predominantly compared to its predecessor PopIns v.1.0.1 [Kehr *et al.*, 2016], Pamir [Kavak *et al.*, 2017] and McCortex [Turner *et al.*, 2018], as they were found to be the only other programs tailored to the classification in chapter 4.1.3 that can be successfully applied to reasonable large data sets.

5.1 Assessment of PopIns2 using simulated data

The simulated data mentioned in this chapter was generated entirely for the purpose of the assessment of PopIns2 [Krannich *et al.*, 2021] and is, as opposed to the real genomic sequence data later in this chapter, not publically available in its entirety. However, chapter 5.1.1 provides a comprehensive description of the simulation process such that an interested user can repeat the evaluation. The data simulation was inspired by the original publication of PopIns ([Kehr *et al.*, 2016], chapter 3.5.1) and thus implements similar concepts. In 5.1.2, the simulated data is used to determine how different setups of the PopIns2 workflow perform in terms of NRS detection and how they compare to other tools.

5.1.1 Data simulation pipeline

The following workflow (Figure 24) is designed to generate many sets of paired short-read sequencing data which, if aligned to a given reference, yield NRS. The reference used for the workflow is the human chromosome 21 (downloaded from `ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA_000001405.15_GRCh38/seqs_for_alignment_pipelines.ucsc_ids/GCA_000001405.15_GRCh38_no_alt_analysis_set.fna.gz` on November 8th, 2016) as

it is the chromosome with the fewest total base pairs of the human reference genome at version GRCh38.

Let I_{sim} be a set of sequences from the chromosome 21. I_{sim} contains 500 randomly selected sequences of mean length 1455.17 with standard deviation of 1526.08, a minimum length of 41 and a maximum length of 9190. Further, every two sequences in I_{sim} are non-overlapping, contain no unidentified bases 'N' and are at least 1000 bp distant from the nearest 'N' sequence in chromosome 21. The sequences I_{sim} were cut from the original human chromosome 21 resulting in a modified chromosome 21 ($chr21^{-}$).

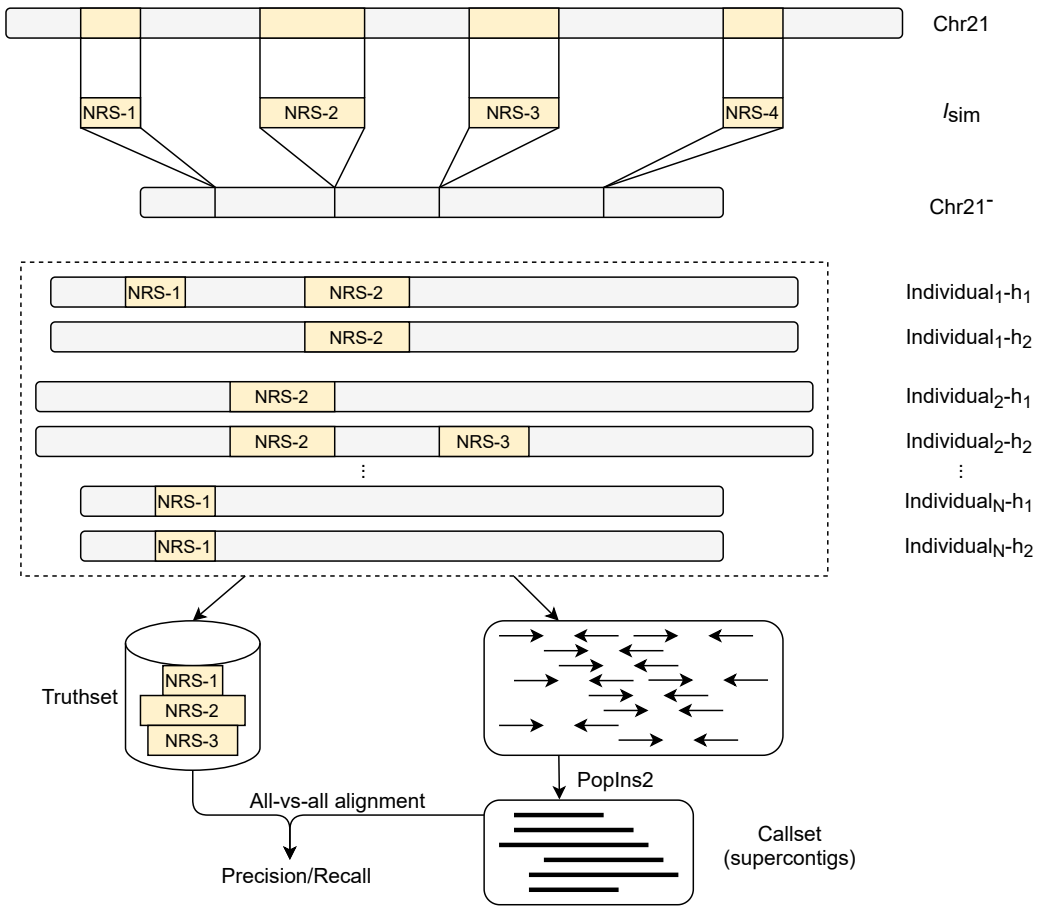

Figure 24: Pipeline for data simulation and callset evaluation. The data simulation of this pipeline results in a set of paired short-reads per simulated individual that, if aligned to a modified reference chromosome, yields NRS (truthset). A set of NRS generated by a SV caller (callset) from the sets of reads is compared to the truthset to determine the precision and recall of the SV detection.

The $chr21^-$ is used to generate data of diploid individuals. First, for each simulated individual two copies (haplotypes) h_1, h_2 are created from $chr21^-$. Next, two sets $I_{h_1}, I_{h_2} \subseteq I_{sim}$ are selected and its sequences incorporated into h_1, h_2, respectively. At this point the two haplotypes h_1 and h_2 together contain the set of sequences $I_h = I_{h_1} \cup I_{h_2}$ as NRS with respect to $chr21^-$. Note that this way of simulating NRS with respect to $chr21^-$ does not add artificially generated sequence to the data simulation pipeline but only keeps operating on real human sequences that originated form chromosome 21. Finally, from the two haplotypes h_1, h_2 reads are generated [Huang *et al.*, 2012] and aligned [Li, 2013] to the reference genome. The generated reads are paired short-reads with the error profile for individual base pairs as with the *Illumina HiSeq 2500* sequencer, 150 bp long, of 15x read coverage per haplotype, generated from 300 bp fragments (standard deviation $\sigma = 50$ bp) of the simulated haplotypes and aligned to $chr21^-$.

Using the described data simulation pipeline, 100 individuals were simulated resulting in 100 sets of reads at 30x total read coverage. The set of reads with their corresponding alignment information of all individuals are used as input data for PopIns2, PopIns and Pamir, the raw sequences are used as input data for McCortex [Turner *et al.*, 2018]. All four tools return a set of NRS across all individuals (callsets).

5.1.2 Evaluation of NRS callsets

The callset $\mathcal{C}$ of each SV calling method is compared to the *truthset* $\mathcal{T}$, which is the union of I_h from all individuals. To compare a callset with $\mathcal{T}$ an all-vs-all alignment was performed, i.e. every sequence $t \in \mathcal{T}$ is aligned to every $c \in \mathcal{C}$. An alignment is reported if STELLAR [Kehr *et al.*, 2011] finds and an alignment with a minimum length of 50 base pairs and a maximum error rate of 0.05.

Once all alignments under the given conditions are computed, a *bipartite matching* procedure (Figure 25) is used in order to determine the number of *true positives* (TP), *false positives* (FP), *false negatives* (FN) and *redundant alignments*. Every $t \in \mathcal{T}$ that is at least 90% covered by at least one individual alignment with a sequence $c \in \mathcal{C}$ is counted as a TP. All remaining $t \in \mathcal{T}$ that were not counted as TP are counted as FN, i.e. $FN = |\mathcal{T}| - TP$. The number of FP is calculated as $|\mathcal{C}| - TP$. This definition of FP implies that in this evaluation is more stringent than prior analysis [Kehr *et al.*, 2016; Rizk *et al.*, 2014] because redundant alignment are counted as FP. The number of redundant alignments is calculated as the number of all $c \in \mathcal{C}$ that cover 90% of a t with an alignment minus TP. From the counts the recall, precision and F_1 score of the NRS detection are calculated as $\frac{TP}{TP+FN}$, $\frac{TP}{TP+FP}$ and $\frac{TP}{TP+\frac{1}{2}(FP+FN)}$, respectively. The F_1 score is the harmonic mean between precision and recall.

To evaluate the new merge method of PopIns2, each set of reads of the 100 simulated individuals got passed to the PopIns2 assemble module to create a set of contigs per individual before passing all sets of contigs to the new PopIns2 merge module. Additionally,

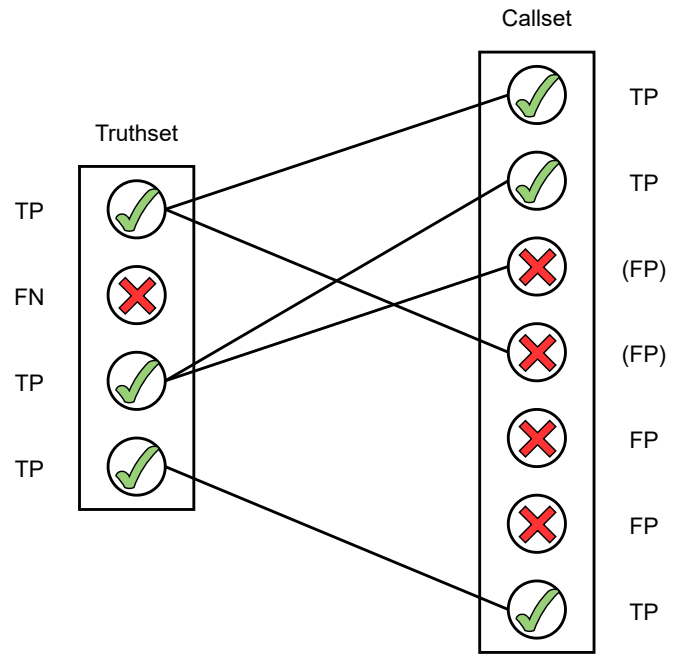

Figure 25: Bipartite matching graph between a truthset and callset of NRS.

the contig assembly of the PopIns2 assemble module was assessed with different external assembly software: Minia [Chikhi and Rizk, 2013] as it is a more recent software, Velvet [Zerbino and Birney, 2008] for backward compatibility with the predecessor PopIns and SPAdes [Bankevich *et al.*, 2012] because it is a widely established software. Another entrenched software [Peng and Xu, 2011] has a seemingly fitting objective to be considered but was quickly ruled out due to its lack of compatibility with the format the aligned reads are stored in.

ASF	Avg. selected reads per sample	Avg. contigs per sample
0.50	19647.20	152.2
0.67	20921.86	145.1
0.75	21358.84	140.2

Table 1: Average number of selected reads and contigs per simulated individual resulting from the read selection and contig assembly of the *PopIns2 assemble* module.

Impact of the ASF parameter on the assembly step. After the read alignment in data simulation pipeline, each individual yields an average of 7.49 million reads where 7.38 million reads are fully or partially aligned to their correct original position in $chr21^{-}$, 12.7 thousand reads are labelled as unaligned and 99.1 thousand reads are fully or partially aligned to $chr21^{-}$ but at a wrong location with respect to their position of origin. With the given alignment information per individual, the ASF parameter (chapter 4.3.5) plays a key role for the number of reads being selected for the contig assembly of the PopIns2 assemble module. A range of practical values for the ASF parameter is shown in Table 1. Among the reads being selected for the contig assembly 35.36%, 41.84% and 44.07% of the reads have a poor alignment to $chr21^{-}$ with ASF parameter setting 0.50, 0.67 and 0.75, respectively. Interestingly, within the scope of tested ASF values there is an inversely proportional relation between the number of selected reads and the number of contigs assembled with Minia (default assembler of the PopIns2 assemble module).

Evaluation of the merging step. The NRS detection of PopIns2, PopIns, Pamir and McCortex was evaluated taking the contig assemblies from the three different ASF parameter settings, using different numbers of individuals (subgroups) and applying the different external assembly methods. The callset of each setup is evaluated for its recall, precision and F_1 score with respect to the corresponding $\mathcal{T}$ (there are multiple truthsets for different numbers of individuals). Pamir reports the detected NRS (in its original publication called *events*) with 1000 bp flanking regions of the reference genome. To avoid that the all-vs-all alignment underlying the calculation of precision and recall is penalized because of the flanking regions (the flanking regions are not part of the $t \in \mathcal{T}$), the flanking regions were trimmed off prior to the evaluation of the callset.

The results from the evaluation are summarized in Table 2 and 3. Among all tested setups the combination of PopIns2 with Minia exhibits the highest precision. Pamir exhibits the higher recall compared to the setups with PopIns and PopIns2. Velvet and SPAdes were discarded from further analysis as early results indicated that Minia performs best in the context of NRS detection with PopIns2. PopIns2 was tested with ASF parameter values 0.5 (default in PopIns), 0.67 (default in PopIns2) and 0.75 (very lenient read selection). The evaluation with the group of 50 individuals showed a total gain in F_1 score of 16 and 18.7 percentage points for PopIns and PopIns2, respectively. The three best performing setups in terms of F_1 score differ by 5.5 and 4.8 percentage points for the group of 50 and 100 individuals, respectively.

A more detailed analysis of the best performing methods and setups is shown in Figure 26. The recall and precision of PopIns, PopIns2 and Pamir were evaluated using an increasing number of individuals. PopIns2 has the best recall for smaller numbers of individuals up to about 20 individuals, then Pamir outperforms the best setup of PopIns2 by 2.2 to 4.7 percentage points. The recall of each setup of PopIns and PopIns2 is virtually constant. Pamir revealed a two and a half times increase in recall from 0.29 to 0.77 within the first 20 individuals before reaching its plateau and steady recall. The precision of PopIns2 is

SV caller	Assembler	#Individuals	ASF	Recall	Precision	F_1 score
PopIns	Velvet	50	0.50	0.623	0.536	0.576
PopIns	Minia	50	0.50	0.591	0.589	0.590
PopIns	Minia	50	0.67	0.715	0.748	0.731
PopIns	Minia	50	0.75	0.742	0.758	0.75
PopIns2	Minia	50	0.50	0.589	0.616	0.602
PopIns2	Minia	50	0.67	0.709	0.781	0.743
PopIns2	Minia	50	0.75	0.734	0.854	0.789
PopIns2	SPAdes	50	0.67	0.673	0.669	0.671
Pamir	-	50	-	0.776	0.837	0.805
PopIns2	Minia	100	0.67	0.708	0.789	0.746
PopIns2	Minia	100	0.75	0.737	0.862	0.794
PopIns2	SPAdes	100	0.67	0.622	0.593	0.608
Pamir	-	100	-	0.784	0.747	0.765
McCortex	-	100	-	0.010	0.833	0.020

Table 2: Precision and recall of NRS callsets from simulated human short-read data.

SV Caller	Assembler	#Individuals	ASF	#elements covered from truthset	|Callset|	#elements being aligned from callset	#elements being redundant from callset
PopIns	Velvet	50	0.50	297	554	298	1
PopIns	Minia	50	0.50	282	479	286	4
PopIns	Minia	50	0.67	341	456	347	6
PopIns	Minia	50	0.75	354	467	370	16
PopIns2	Minia	50	0.50	281	456	281	0
PopIns2	Minia	50	0.67	339	432	341	2
PopIns2	Minia	50	0.75	350	411	355	5
PopIns2	SPAdes	50	0.67	321	480	337	16
Pamir	-	50	-	370	442	418	48
PopIns2	Minia	100	0.67	347	440	349	2
PopIns2	Minia	100	0.75	361	419	365	4
PopIns2	SPAdes	100	0.67	305	514	323	18
Pamir	-	100	-	384	514	480	96
McCortex	-	100	-	5	6	5	0

Table 3: Counts of the precision and recall statistics of Table 2. The truthset for the group of 50 and 100 individuals contains 477 and 490 contigs, respectively.

relatively constant at 0.77 to 0.86 depending on the setup with the exception of a slight drop of up to 4 percentage points at about 15 to 20 individuals. PopIns2 shows the best precision for all data sets of more than 40 individuals (precision of 0.87) with up to 6.6 percentage points over Pamir. Pamir has its highest precision at very few individuals and shows a slowly linear decline in precision with an increasing number of individuals. The

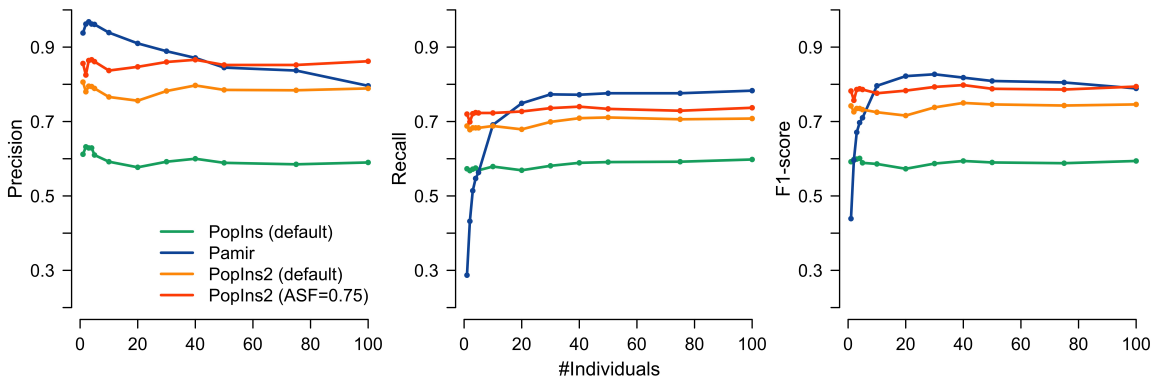

Figure 26: Evaluation of the SV detection with PopIns, PopIns2 and Pamir using simulated data. Both PopIns and PopIns2 were set to use Minia as external assembler for the individuals' contig assemblies but used their default settings for the ASF value (0.50 for PopIns and 0.67 for PopIns2) if not stated otherwise. For each software precision, recall and F_1 score are shown depending on an increasing number n of individuals given as input data ($n = 1, 2, 3, 4, 5, 10, 20, 30, 40, 50, 100$).

linear decline in precision is predominantly attributed to the rapid increase of redundant sequences in the callsets of Pamir (numbers shown in Table 3).

For each SV caller the TP among the NRS callsets were categorized by their length in base pairs to verify whether the detection has a bias towards certain contig lengths. However, the length distributions resulting from all tested programs are approximately proportional to the length distribution of the truthset (of 50 individuals). Pamir's callset has a few more NRS in accordance with the truthset for the length categories smaller than 1000 bp while PopIns2 has a few more NRS in the length categories from 1000 to 5000 bp. It is important to note here that PopIns2 does not report NRS smaller than $2k - 1$ due to (the recommended) graph simplifications of Bifrost. All tests of PopIns2 using simulated data were performed with $k = 63$.

5.1.3 Preliminary results utilizing the multi-k module

In chapter 4.3.4 a multi-k algorithm for the construction of a CDBG was introduced. As stated earlier, this method is still experimental and not part of the official release build of PopIns2, yet. Nevertheless, preliminary results using the simulated data indicate the potential of this method.

The multi-k method is implemented in a separate module of PopIns2. To assess and compare the precision and recall of the method the *PopIns2 multik* module was utilized to compute a CDBG from the sets of unaligned and poorly aligned reads for all growing

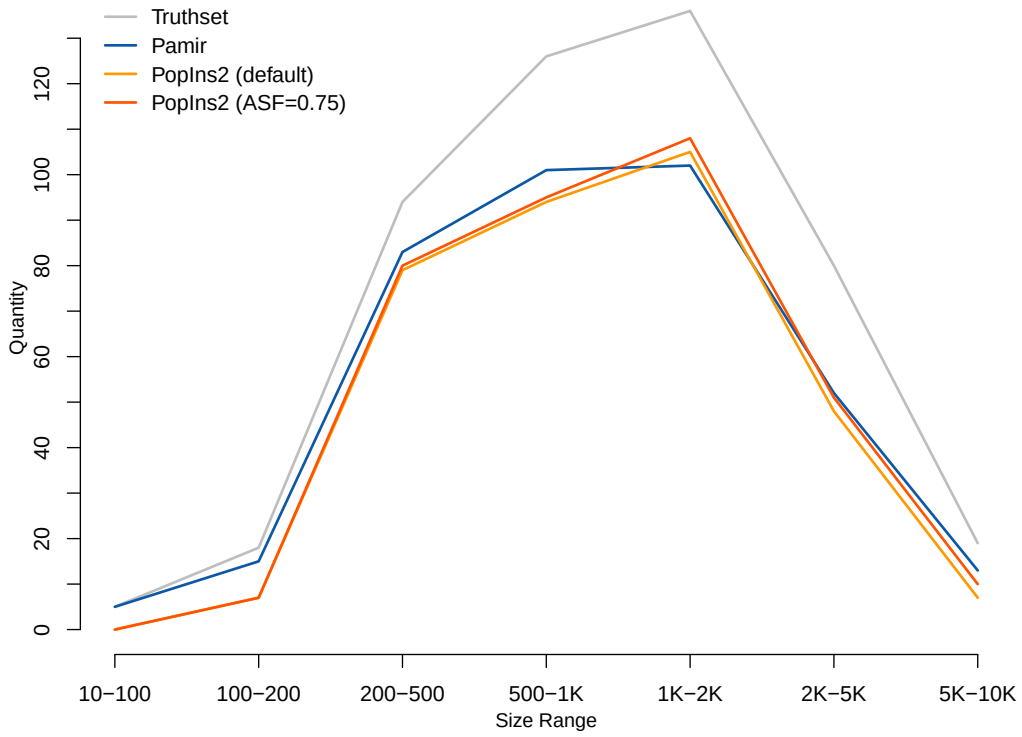

Figure 27: Number of TP from the NRS callsets separated by different size ranges. The grey line shows the ground truth (numbers from truthset). The colored lines show the results for PopIns2 and Pamir. The analysis was performed with the simulated data of 50 individuals.

subsets of individuals as in Figure 26. The iteration was set to start with an initial k-mer size $k = 27$ growing by the step size $\Delta_k = 20$ until it reaches the maximum k-mer size $k_{max} = 127$. Subsequently, the resulting CDBGs were used as input data for the merge module (using parameter $-y$ for the GFA file, $-z$ for color file and $-k$ to match the k-mer size of 127 where the multi-k module stopped). For comparison with Table 3, the computation of supercontigs using the final CDBG that was created from the multi-k module and the full set of 100 individuals resulted in 479 supercontigs.

Even though the supercontig callsets resulting from the workflow including the multi-k module seem promising at first, e.g. with the simulated data of all 100 individuals the cardinality of the callset has the smallest absolute deviation from the truthset among Pamir and the best performing setups of PopIns2 without the multi-k module, there is a conspicuous tendency in the tests with larger sets of individuals. Figure 28 shows an excerpt of the results from PopIns2 with ASF=0.75 and Pamir again (as they were the

best performing software and setup) together with the results from the PopIns2 workflow including the multi-k module. The PopIns2 workflow including the multi-k module has a decent performance up until an input of 80 individuals improving the precision and recall of PopIns2 across all sizes of input sets larger than five individuals. However, what raises concern at the present state of the method is the decline in recall and an even steeper decline in precision for the NRS detection if more than 80 individuals are provided as input data. The test with 100 individuals resulted in a worse F_1 score than the test without the usage of the multi-k module and Pamir. As PopIns2 aims to process much larger numbers of individuals in practise this has to be subject to further analysis.

5.2 Application of PopIns2 using the reads of many human individuals

As opposed to the last chapter, the following evaluations of PopIns2 utilize real human short-read sequencing data. Real sequencing data is typically a lot more complex than simulated data, i.e. both NRS detection and genotyping methods have to deal with more sequencing artifacts as well as more repetitive sequences. Moreover, the following applications utilize a much larger number of individuals with a much larger amount of read data. Together, solving NRS detection and genotyping on real genomic sequences as well as very large quantities of data are ultimately the real challenge and objective of PopIns2.

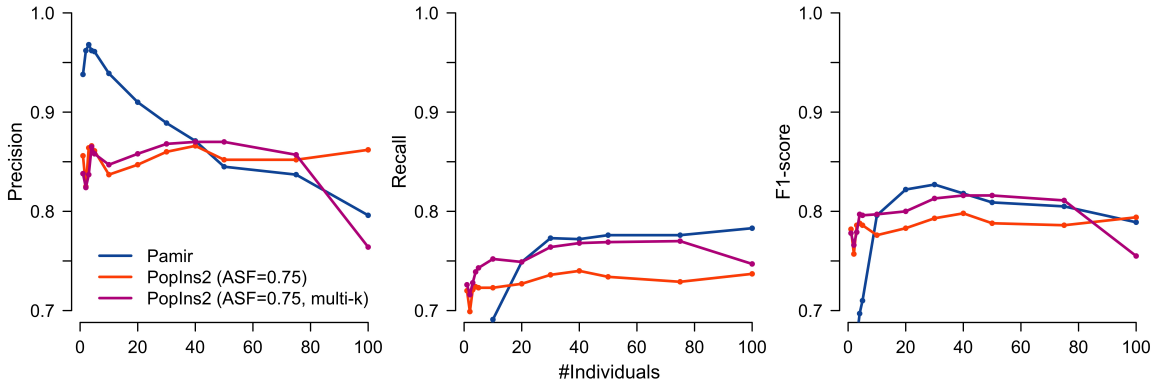

Figure 28: Evaluation of the SV detection with PopIns2, PopIns2 with the multi-k module and Pamir using simulated data. For each software precision, recall and F_1 score are shown depending on an increasing number n of individuals given as input data ($n = 1, 2, 3, 4, 5, 10, 20, 30, 40, 50, 100$).

5.2.1 Detecting NRS in the Polaris Diversity Cohort

To assess the new merge module on real population-scale data sequencing data PopIns2 was applied to the Polaris Diversity Cohort (PDC). The PDC comprises short-read sequencing data (*Illumina HiSeqX* platform) of 150 individuals from three continental groups (AFR, EUR, EAS) at a targeted sequencing coverage of 30x per individual. Since PopIns, PopIns2 and Pamir all require aligned reads as input data, the reads of each individual were aligned [Li, 2013] to the human reference (hg38) prior to the analysis. Pamir was not thoroughly assessed in this analysis because the program execution exceeded the longest possible computation time of 28 days using 16 threads on the HPC cluster in use [Krannich *et al.*, 2021].

The assembly step. The aligned reads (BAM file format) are used as input data for the PopIns2 assemble module. PopIns2 assemble with default parameters produced an average of 8049 contigs per individual from about 1.2 million reads that were classified as unaligned or poorly aligned to the reference genome. Later, for a fair comparison these sets of contigs were subsequently used for both the PopIns and PopIns2 merge module. The contig assembly reduces the disc memory requirement from an average 1.6 GB for the selected reads to 6 MB for the set of contigs.

The individual program executions (*instances*) of PopIns2 assemble can be effectively distributed across a HPC cluster environment and support multi-threading CPU architecture for the computationally most intensive tasks (decontamination and assembly). For the analysis of the PDC the PopIns2 assemble instances were distributed across 16 compute nodes (Intel(R) Xeon(R) Gold 6130 CPU @ 2.10 GHz) that support the same CPU instruction set, s.t. SIMD instructions stayed enabled (default). The individual instances took approximately 80 hours wall clock time to finish.

The merging step. The sets of contigs from the PopIns2 assemble module were used as input data for the PopIns2 merge module. The merging of the contigs resulted in 15306 supercontigs, generated in 50 minutes CPU time using a single thread (total wall clock time was 139 minutes, see Table 4). Building and traversing the CDBG are the dominating factors for the total computation time with a ratio of approximately $1:3$, respectively. In comparison, using the same contigs as input data for PopIns, 13456 supercontigs were reported in 94 minutes CPU time using a single thread. Figure 29 shows the length distributions for the sets of supercontigs generated with PopIns and PopIns2. Both callsets exhibit similar length distributions with characteristic peaks at the 300 base pairs mark which is known to be caused by *SINEs/Alu elements* [Deininger, 2011; Ade *et al.*, 2013] in the human DNA. Moreover, the right panel of Figure 30 shows a monotonic increase in the number of supercontigs generated with PopIns and PopIns2 merge for a growing number of individuals (i.e. the number of contig sets). Interestingly, one can clearly observe a jump at the 100 individuals mark. The data from the PDC is sorted by continental groups and therefore at each 50 individuals a new continental group is added to the merging process.

The jump at the 100 individuals mark is due to the additional continental group (AFR) being introduced to the merging process at this mark.

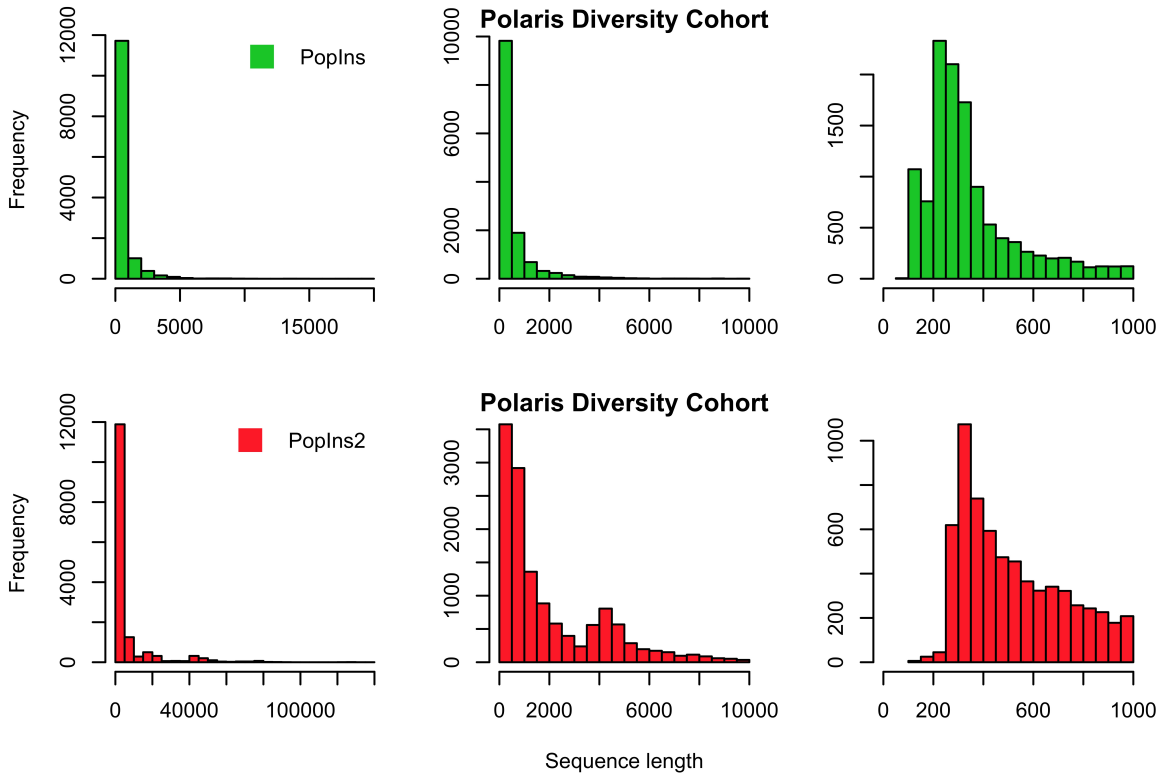

Figure 29: Length histograms of the supercontigs from the PDC generated with PopIns and PopIns2. Every panel shows the same data but in different zoom levels on the x-axis.

Function	Wall clock time [min]
Graph build (threads)	35 (1)
Graph simplification	3
Graph color annotation	2
Graph traversal	98
Other	1

Table 4: Wall clock times measured in minutes for PopIns2 merge computing supercontigs from the PDC.

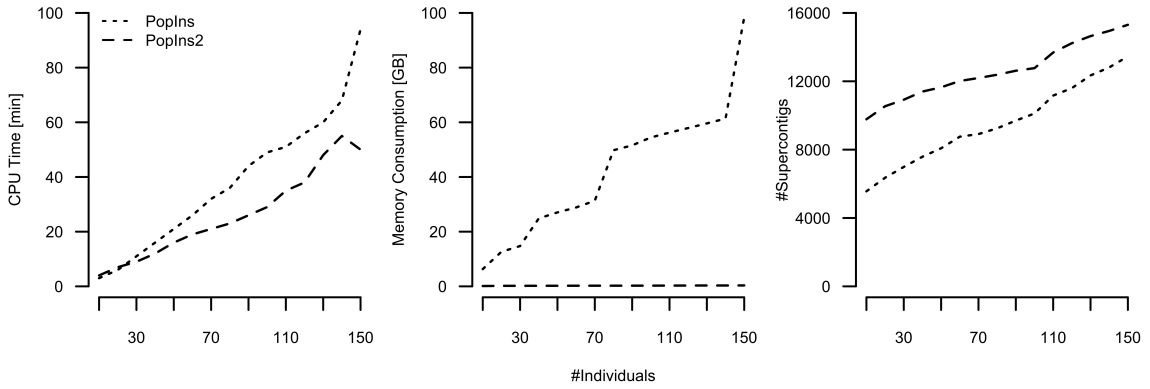

Figure 30: Benchmarks of the PopIns and PopIns2 merging modules using a growing number n of contig sets of individuals from the PDC ($n = 1, 2, 3, 4, 5, 10, 20, 30, 40, 50, 100$). The left panel shows the CPU time in minutes, the middle panel shows the main memory consumption in gigabytes and the right panel shows the number of supercontigs.

Further, the merge algorithms of PopIns and PopIns2 were examined for their scalability by analysing their required CPU time and main memory consumption for a growing number of individuals (left and middle panel of Figure 30). PopIns shows a rapid increase in main memory consumption for growing numbers of individuals. Merging the contigs of the entire PDC with PopIns requires almost 100 GB of main memory. In contrast, PopIns2 requires only 342 MB of main memory for merging all 150 sets of contigs and can store the corresponding CDBG in only 154 MB disc space. Within the scope of tests performed using the PDC, both PopIns and PopIns2 show a computation time that grows linearly with the number of individuals. However, the computation time of PopIns2 grows substantially slower than that of PopIns.

Figure 31 shows two heatmaps which display the effect of the DBG parameter k (k-mer length) and g (minimizer length) on the computation time and memory consumption of PopIns2 merge in more detail. The upper heatmap indicates an inversely proportional relationship of the minimizer length to the computation time, i.e. the larger g the lower the CPU time. These observations follow the expectations that the choice of a larger minimizer length results in fewer lookup operations (into the index structure of Bifrost, see 2.6) in order to determine whether a k-mer is a substring in one of the unitigs. The lower heatmap shows that there is a favourable middle ground between k-mer and minimizer length to optimize the memory consumption. A ratio of approximately $1.5 : 1$ between k-mer and minimizer length appears to be a sensible setting. Therefore, $k = 63$ and $g = 47$ were chosen as defaults for PopIns2 merge.

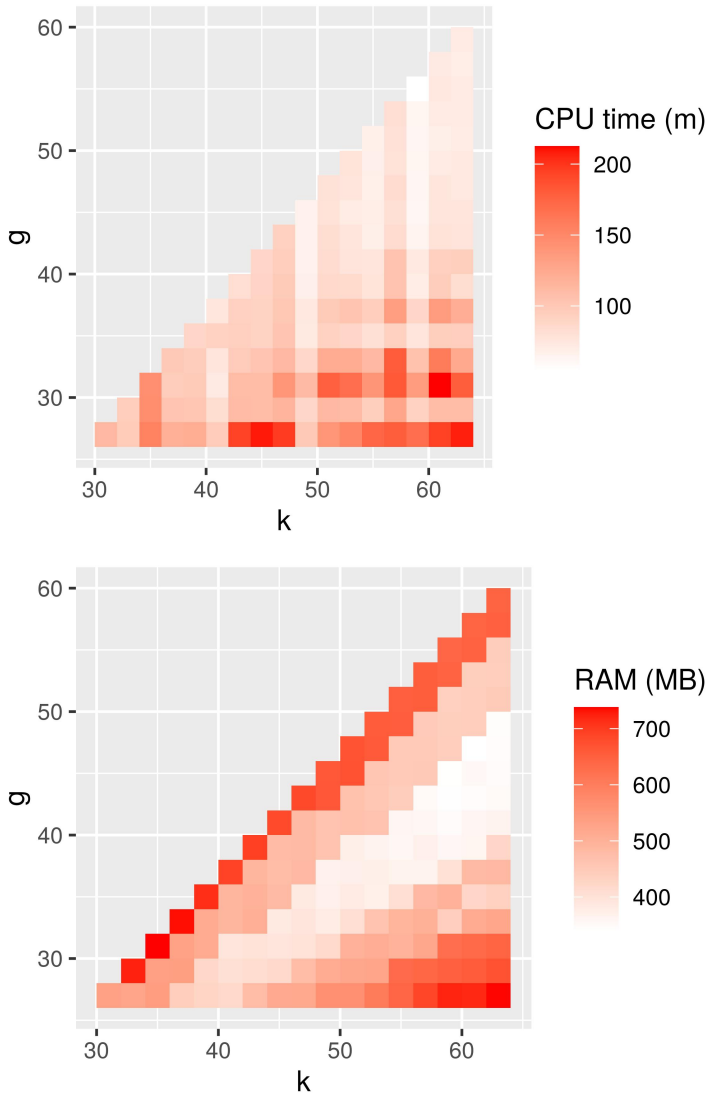

Figure 31: Parameter space exploration of kmer size (k) and minimizer size (g). Both parameters g and k influence the construction time, k-mer accession time and memory requirement of the colored de Bruijn Graph and hence the performance of PopIns2 merge. The upper heatmap shows the CPU time measured in minutes for PopIns2 merge. The lower heatmap shows the peak memory consumption during the computation of PopIns merge in megabytes. A high (undesired) and low (desired) demand of computational resources are denoted in red and white, respectively. Each tile in the heatmaps is the result of a parameter combination of odd values of k and g, i.e. $k = 31, 33, ..., 63$ and $g = 27, 29, ..., 59$. Bifrost, and consequentially PopIns2 merge, is constrained to odd values of k, odd values of g and $g < k - 1$.

5.2.2 Genotype assessment using the Polaris Kids Cohort

In addition to the PDC the Polaris project comprises short-read data of another study group called the *Polaris Kids Cohort* (PKC). The 50 individuals of the PKC are the F_1 offspring of 100 parent pairs of the PDC. As of March 22nd, 2021, 49 individuals of the PKC were publically available such that 49 family trios were utilized for NRS detection and genotyping. The final genotype predictions of PopIns2 are assessed by their concordance with Mendelian Inheritance rules (chapter 2.3) and expected allele transmission rates.

Running PopIns and PopIns2 on the 49 trios. Analogous to the previous analysis in 5.2.1, the read pairs of the PKC were aligned to the human reference genome and used as input data for the PopIns2 assemble module. PopIns2 assemble generated on average 8187 contigs per individual. Next, the 49 contig sets of the PKC together with the corresponding 98 contig sets of the PDC were used as input data for the PopIns and PopIns2 merge modules. PopIns and PopIns2 generated sets of 12889 and 15450 supercontigs, respectively. Figure 32 shows the overlap between the sets of supercontigs. The two sets overlap each other by 19-65% depending on the chosen threshold for a minimum sequence overlap by sequences of the orthogonal callset.

Finally, the placing and genotyping modules were applied to the callset of both PopIns and PopIns2. The genotype predictions for all 147 individuals were jointly reported in one multi-sample VCF file using VCFtools [Danecek *et al.*, 2011].

Running Pamir on the 49 trios. As the 49 family trios and the PDC itself comprise approximately the same number of individuals and read depth it is assumed that running Pamir with the entire set of trios at once would not finish its computation within the maximum computation time of the given HPC cluster. Therefore, Pamir was applied to one trio at a time, i.e. Pamir computed 49 callsets from three individuals each.

Variant counts. The SV calling from the 49 trios resulted in a median of 2256, 2463 and 1873 NRS per individual for PopIns, PopIns2 and Pamir, respectively. Figure 33 shows the number of NRS sequences per individual separated by the continental groups. The NRS were counted in the final VCF files of insertions, i.e. every counted NRS was partially or fully placed in the reference genome and genotyped. Further, a NRS was only counted if it has a minimum length of 50 bp and its corresponding insertion is not genotyped as homozygous reference (0/0). Consistent with previous studies [Wong *et al.*, 2020; Abel *et al.*, 2020] the individuals from the African superpopulation (AFR) reveal the highest average numbers of NRS. Since there is no unified NRS callset from all 49 trios available for Pamir, the accordance of NRS between all three pairwise combinations of SV calling methods was compared for each trio separately (Figure 34). PopIns and PopIns2 showed the largest absolute intersection of NRS among those three pairs. The median number of NRS per trio is 1650, 1898 and 6661 for PopIns, PopIns2 and Pamir, respectively.

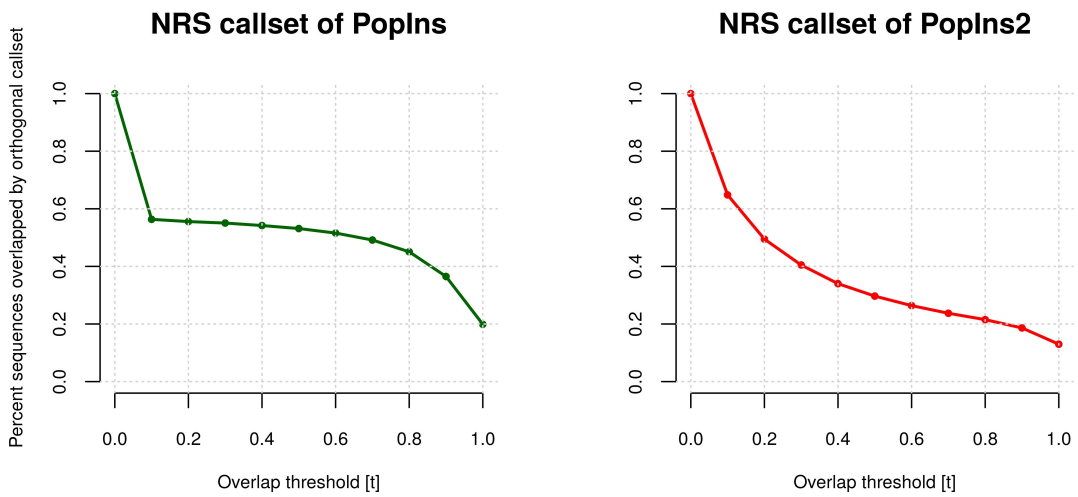

Figure 32: Sequence overlaps between the callsets of PopIns and PopIns2 merge utilizing the 147 related individuals of the PDC and PKC. The two panels show how many sequences of the callset of PopIns (C_{P_1}) have a fractional sequence overlap with a sequence of the callset of PopIns2 (C_{P_2}) and vice versa. To determine the sequence overlaps an all-vs-all alignment between C_{P_1} and C_{P_2} was computed [Kehr *et al.*, 2011] using standard parameter (minimum alignment length of 50 and maximum error rate of 0.05). Here, a supercontig $s_1 \in C_{P_1}$ is *overlapped* by a supercontig in $s_2 \in C_{P_2}$ (and vice versa) if one or multiple local alignments of s_2 span a fraction of a least t base pairs in s_1. For both tools the fraction of supercontigs overlapped by the orthogonal callset was tested for the thresholds $t = 0.1, 0.2, ..., 1.0$.

Principal component analysis. In preparation for the subsequent analysis the genotype predictions of PopIns2 were subject to a sanity check. If the genotypes are fairly accurate then a Principal component analysis (PCA) should be able to distinguish the individuals by their SVs and to cluster them by their corresponding continental groups. Analogous to the procedure in [Niehus *et al.*, 2021] variants with the exact same genotypes for all individuals and variants in linkage disequilibrium were excluded from the variants that are utilized for the PCA as they are not or weakly informative to distinguish the individuals. After the filtering the remaining 1787 variants from the final VCF file of PopIns2 were used for a PCA (Figure 35). The first and second principal components distinctly cluster the continental groups with 5.138% and 2.217% explained variance, respectively.

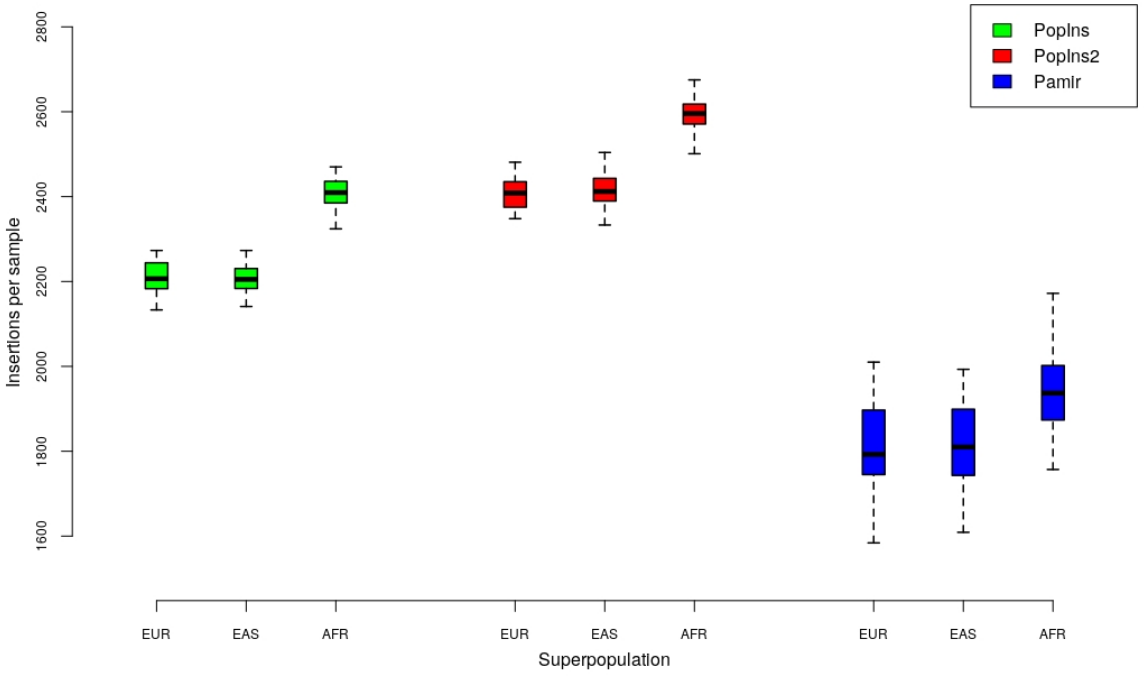

Figure 33: Number of NRS per individual of the 49 trios separated by continental groups (superpopulations).

Mendelian inheritance error rate and transmission rate. So far the majority of evaluations assessed and quantified the NRS detection phase of the SV calling methods. However, another quality measure for a callset of germline NRS is that the variant genotypes comply with the Mendelian inheritance rules and expected transmission rate. Since there is no truthset available for real data sets, deviations from these rules or expectations aid the identification of NRS with spurious genotypes that the user might wants to separate from the higher quality results. As PopIns and PopIns2 both report genotypes and their corresponding genotype qualities for every NRS, the predicted genotypes can be assessed using the pedigree information. Pamir will be subject to a slightly different but comparable assessment since Pamir does not report genotype qualities or likelihoods for its predicted genotypes.

The Mendelian inheritance error rate and transmission rate was calculated as in [Niehus *et al.*, 2021] for both PopIns and PopIns2. The Mendelian inheritance error rate is an indicator how credible the genotype predictions are. It is the fraction of genotypes in the offsprings that cannot be explained with the Mendelian inheritance rules and the parental genotypes. The transmission rate is the frequency how often an allele is inherited from parent to child generation. In the diploid human genome it is expected that a

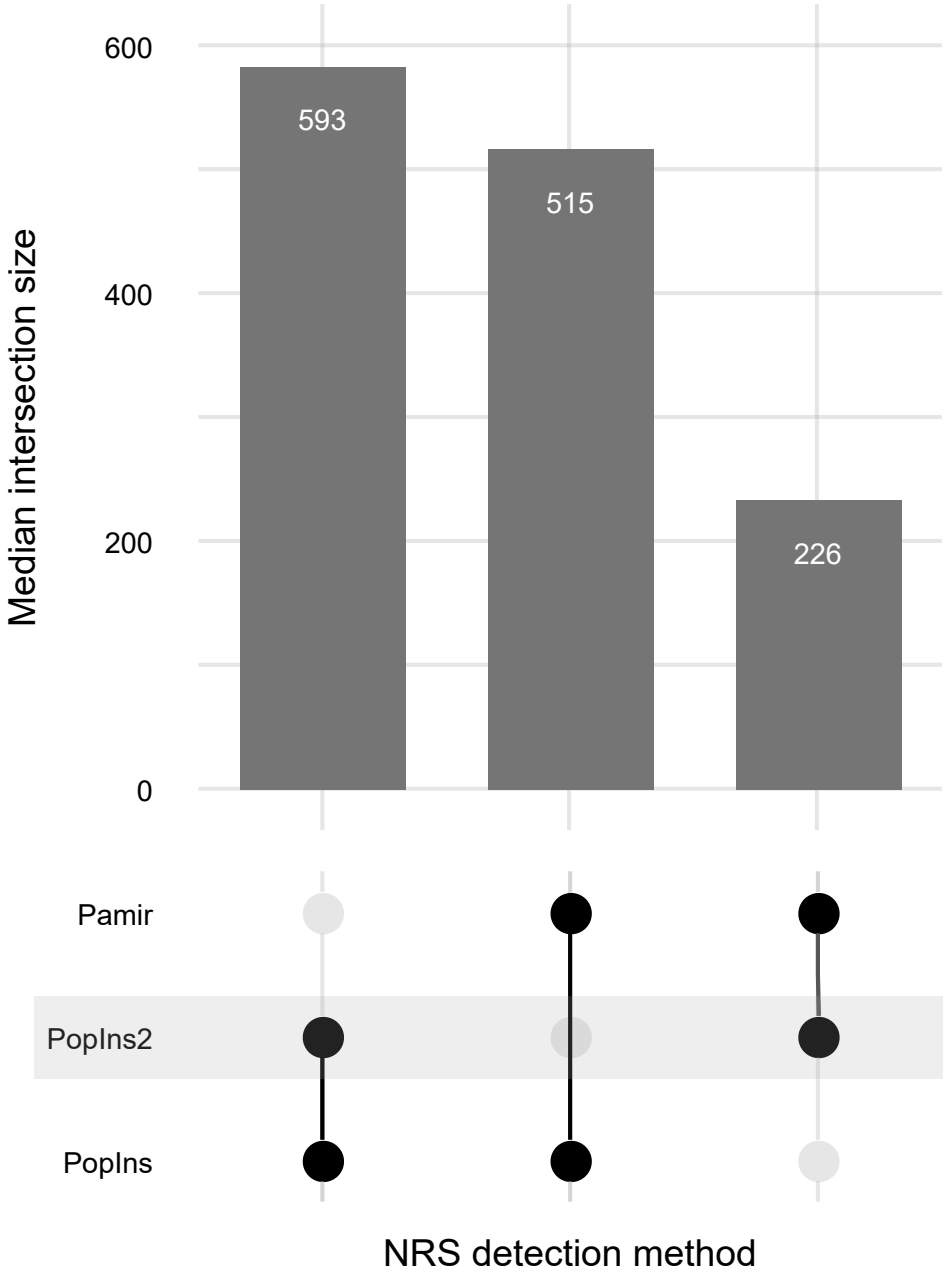

Figure 34: Median absolute intersection of NRS per trio for all pairwise combinations of SV callers. The intersection of two NRS callsets A, B is the number of distinct pairs (a, b) of sequences $a \in A, b \in B$ which align with a reciprocal sequence overlap of at least 50% (using the same software and parameters as in Figure 32). For a fair and comparable alignment among all SV callsets the NRS of Pamir had their flanking reference sequences trimmed. NRS were considered for alignment only if they are partially or fully placed in the reference genome, have a minimum length of 50 bp and their corresponding insertion is not genotyped as homozygous reference (0/0).

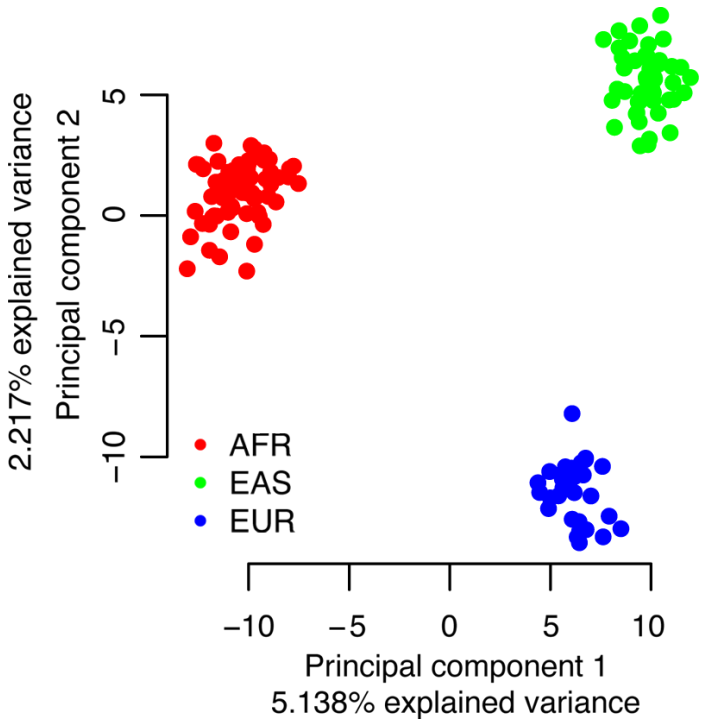

Figure 35: Principal component analysis (PCA) using PopIns2's NRS genotype predictions of the trio data. The NRS genotypes of all individuals were converted into a matrix $M^{V \times I}$, where V is the number of variants and I the number of insertions. Duplicated variants (rows of M) and variants with a significant Spearman correlation were removed from M. Finally, 1787 remaining informative variants were used for the PCA. The red, blue and green data points encode the African, European and East Asian superpopulations, respectively.

heterogeneous carrier of a variant transmits the variant allele in 50% of the cases. In this analysis the transmission rate was determined from all cases where one parent is a heterozygous carrier and the other parent is a homozygous non-carrier. Additionally, since heterozygous variant calls were found to be particularly overabundant among predictions with low genotype quality, the same assessments of Mendelian inheritance error rate and transmission rate were conducted with only a subset of the NRS callsets where variants are in Hardy-Weinberg Equilibrium (HWE, chapter 2.3).

Figure 36 shows the results of the genotypes assessment for the insertion callsets of PopIns and PopIns2. The analysis of the Mendelian inheritance error rate shows that the callset of both tools can be filtered to a conservative callset of at most 1% error rate. For both conditions (filtered and unfiltered) the number of insertions consistent with the Mendelian inheritance rules is virtually identical. However, the HWE filtered subsets show that

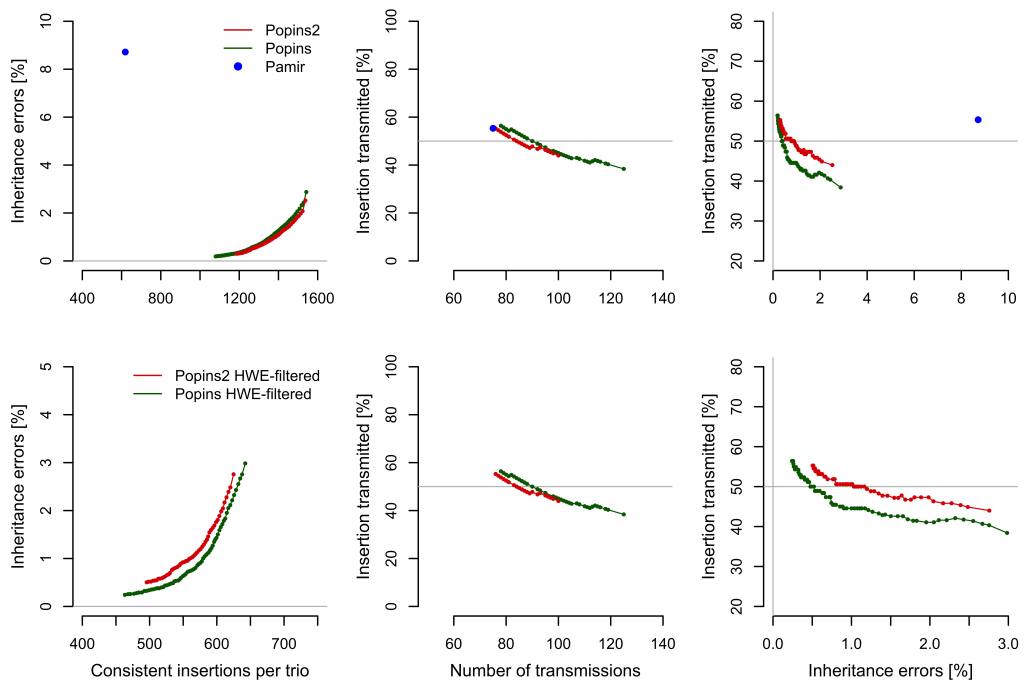

Figure 36: Mendelian inheritance patterns and transmission rate for the insertion genotypes of the 49 Polaris trios. The top row shows the results for the full set of variants, the bottom row shows the results for only the variants that are in Hardy-Weinberg Equilibrium. The left panels show the Mendelian inheritance error rate by the number of insertions per trio that are consistent with the Mendelian inheritance rules. The middle panels show the transmission rate of insertions unique to one trio with one heterozygous parent by the number of overall transmissions. The right panels show the transmission rate of insertions unique to one trio with one heterozygous parent by Mendelian inheritance error rate. Each data point in the panels corresponds to an observation of decreasing minimum genotype quality (from left to right on the x-axis). The grey lines denote the ideal values.

PopIns has a marginally lower (≤ 0.25 percentage points) Mendelian inheritance error rate and reports more consistent insertions than PopIns2. Interestingly, the marginal tendency in terms of the Mendelian inheritance error rate inverts for more lenient genotype quality thresholds, i.e. PopIns2 has a sightly lower Mendelian inheritance error rate. For both the unfiltered and HWE filtered condition PopIns2 has a slightly lower absolute deviation from the targeted 50% transmission rate but reports fewer total observed transmissions.

For comparison the genotypes reported with Pamir undergo the same assessments as with PopIns and PopIns2. However, since Pamir only reports final genotype predictions but no corresponding likelihoods or genotype qualities there is only one data point (blue) added to the upper panels of Figure 36. As a reminder, the results of Pamir were computed

using one trio at a time. The data points in Figure 36 are the median values over the individual measurements for each trio. There is no data for Pamir in the lower panels because the three individuals of one trio are not enough data to compute a HWE. The Mendelian inheritance error rates determined for Pamir have median value of over 8.5% which is more than twice the percentage points of any observation from the results of PopIns and PopIns2. The distributions of observed values with Pamir are shown in Figure 37.

5.2.3 Detecting NRS in 1000 Icelandic genomes

In 4.1.1, it was elucidated that the motivation, objective and major challenge for the development of PopIns2 was the scalability, i.e. to enable the NRS detection for a greater number of individuals simultaneously. In this third and last evaluation with real human sequencing data the scalability of the new merge algorithm is put to a test by utilizing a previously prohibitive number of individuals [Kehr *et al.*, 2017]. Here, PopIns2 merge is applied to a data set of 1000 Icelandic individuals [Gudbjartsson *et al.*, 2015*a*; Jónsson *et al.*, 2017].

Previously, the raw NGS data had been processed routinely by the PopIns assemble module such that a set S of unaligned and poorly aligned reads had already been selected per individual. In order to obtain the favourable set of contigs from PopIns2 over PopIns

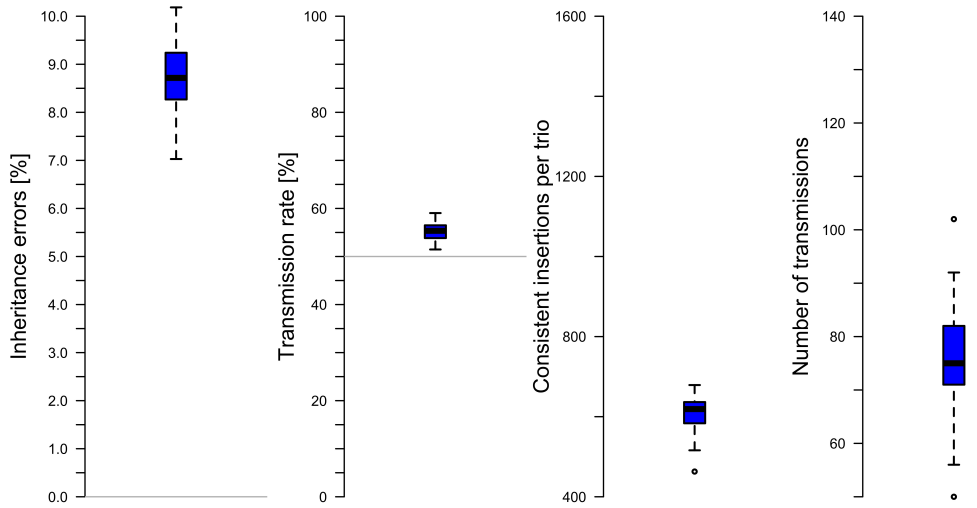

Figure 37: Mendelian inheritance patterns and transmission rate for the insertion genotypes of the 49 Polaris trios. The panels show the distributions resulting from runs of Pamir with one family trio each. Grey lines denote ideal values. The axes describe the same dimensions as in Figure 36.

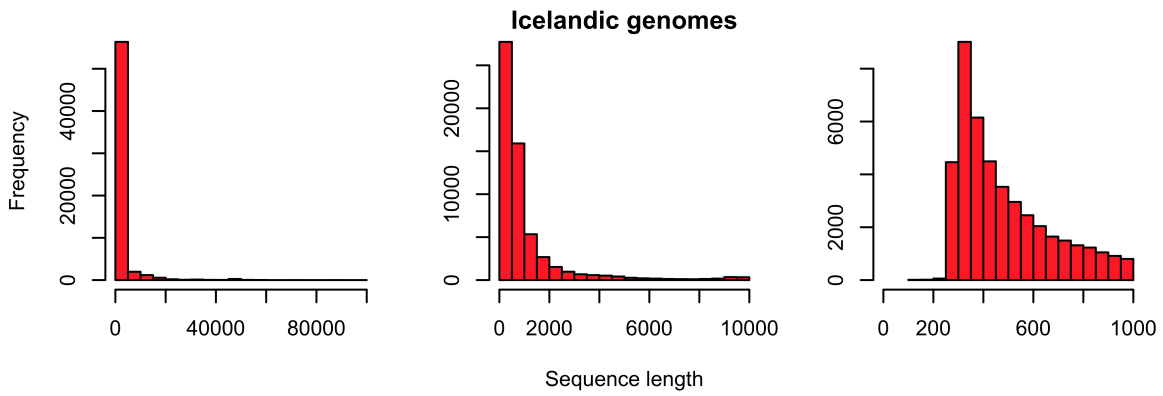

Figure 38: Length histograms of the supercontigs from the 1000 Icelandic genomes cohort. Every panel shows the same data but in different zoom levels on the x-axis.

Function	Wall clock time [min]
Graph build (threads)	3 (24)
Graph simplification	1
Graph color annotation	1
Graph traversal	220
Other	1

Table 5: Wall clock times measured in minutes for PopIns2 merge computing supercontigs from the 1000 Icelandic genomes.

(as shown with the simulated data in chapter 5.1.2) the set S per individual was manually assembled with Minia [Chikhi and Rizk, 2013]. The contig assembly resulted in an average of 6301 contigs per individual. The 1000 sets of contigs were taken as input data for the PopIns2 merge module resulting in a set of 61,515 NRS. The length distribution of the NRS is shown in Figure 38.

The computation of the PopIns2 merge module took 4 hours and 45 minutes wall clock time (3 days, 18 hours and 1 minute CPU time) using 24 CPU core. The contributions to the total computation time are shown in Table 5. As shown in the analysis for the PDC (chapter 5.2.1), building and traversing the CCDBG are the dominating factors for the total computation time. The peak memory requirement during the computation was 2.47 gigabytes of main memory.

6 Conclusion and Future Work

This final chapter summarizes and concludes the presented work on the detection of NRS from population-scale NGS data. The benefits and limitations of the presented method will be discussed and put into context of today's development and direction of the field. Finally, a perspective on potential future extensions to this method and its application will be provided.

Summary

This thesis introduced variants in the human genome as a major driver for environmental adaptation [Norman *et al.*, 2017] and medical conditions. Today, improving medical surveillance and genomic data aggregation continues to determine structural variants as a root cause for severe and sometime rare diseases [Wilfert *et al.*, 2021]. Breakthrough technologies like NGS have greatly improved our ability to sequence large amounts of individuals in a much shorter period of time and to make variant detection a routine analysis. However, detecting each different class of large structural variants from NGS data has comprised several algorithmic challenges. Non-reference sequences, large genomic sequences absent from a given reference genome, are a less investigated and particularly difficult to detect class of SV as they inevitably require de novo assembly. De novo assembly becomes even more difficult, often impossible, if the sequencing data yields a low average genome coverage [Alkan *et al.*, 2011]. Unfortunately, larger sequencing projects comprising up to hundreds or thousands of individuals [McVean *et al.*, 2012] historically have tended to compensate a higher yield per cost with a lower average genome coverage per individual. Thus, NRS detection and genotyping from a single such individual has a lower chance of success or being comprehensive [Kavak *et al.*, 2017]. The limitation of analyzing one individual at a time motivated the development of methods [Kehr *et al.*, 2016; Kavak *et al.*, 2017] which utilize the large quantity of sequenced individuals jointly during the NRS detection. Both most eminent methods specifically tailored for the detection of NRS from the NGS data of many individuals, PopIns and Pamir, were assessed in terms of assembly quality, consistency of the genotypes and computational performance. Each method revealed its superiority in a subset of those aspects.

An ongoing trend of national cohorts and other population-scale sequencing projects as well as recent practical implementations of efficient data structures eventually motivated the development of another NRS detection method scaling to hundreds or thousands of individuals while maintaining high precision and recall. This thesis presents the results of this development, PopIns2, in unprecedented extend and detail. While building on the established predecessor PopIns, PopIns2 introduces an entirely novel approach to generate a NRS callset jointly from the contigs of many individuals. The novel approach first utilizes a fast and highly memory-efficient software [Holley and Melsted, 2020] to construct a colored and compacted de Bruijn Graph. The key idea is that PopIns2 approximates a weighted minimum path cover given the graph and a set of constraints in order to generate a path. Each generated path represents an NRS present in at least one of a population's individuals. PopIns2 has a greatly improved computational performance compared to PopIns and Pamir, i.e. it can process the same numbers of individuals in less time and main memory. Additionally, evaluations on simulated data showed that the NRS callsets from PopIns2 rank among the most precise and sensitive.

PopIns2 is a freely available C++ program under the permissive GNU GPL v2 license. All its dependencies are again non-commercial and freely available. The modular design of PopIns2 facilitates the evaluation of intermediate steps and results as well as the integration of alternative external software for particular tasks of the workflow. The full workflow of all PopIns2 modules has only minimal sequential dependencies and can be trivially automated with a workflow management system and effectively distributed among a HPC cluster environment.

Conclusion

Even though the benefits of genomic data acquisition and analysis at population-scale is known to yield tremendous potential for a future personalized medicine [Taylor *et al.*, 2015; Boycott *et al.*, 2017; Splinter *et al.*, 2018] it requires an ongoing effort to analyze this staggering amount of data. To be able to understand the cause and to develop treatments for cancer, Mendelian diseases, rare and to date unknown diseases we have to reveal their genetic origin and must be able to routinely identify them among a vast quantity of information. The methods and software in this thesis shall provide a contribution to this global scientific effort. PopIns2 provides a practical solution to detect one of the many potentially disease causing structural variants in the genome, the non-reference sequences, from the whole-genome sequencing data of many individuals. Moreover, the algorithmic approaches and analyses shall provide a stimulus for ideas and hint about pitfalls when wielding population-scale data and detecting NRS.

PopIns2 was carefully evaluated on a variety of simulated and real human data sets. Evaluations on simulated data demonstrated that the accuracy of PopIns2 meets that of previous methods while real data shows that the new merging approach scales to orders of magnitude more input data.

The selection of unaligned and poorly aligned reads from each individual is crucial for the assembly of the NRS. Raising the ASF from PopIns' default value led to a substantial increase in precision and recall. Otherwise, a very lenient selection of unaligned reads leads to a distortion of the contig assembly, particularly at the flanking regions of the contigs. The new ASF default value in PopIns2 is adjusted to a reasonable middle ground.

The new merging method in PopIns2 allows simultaneous processing of many genomes together. It utilizes a highly efficient implementation of a CDBG and heavily relies on the color information. The paths generated from the CDBG are eventually translated into NRS being present in one or multiple of the originally many genomes. There exists a strong interplay between the number of genomes being processed and the graph complexity. Fewer individuals can lead to a poor decision making when traversing the CDBG and hence lead to flawed assemblies of the NRS while many individuals lead to more complex graph structures. Still, it was observed in the simulated data that the precision and recall of the NRS detection remains robust with growing numbers of individuals suggesting that the color-based decision making counteracts graph complexity. Moreover, PopIns2 shows the lowest redundancy in its callsets among the tested approaches, i.e. the callsets comprise the fewest cases where multiple sequences from the callset align to one particular sequence from the truthset. The NRS detection with PopIns2 indicates no bias towards certain variant lengths.

Merging NRS using PopIns2 and real human sequence data was performed on a group of 150 and 1000 individuals simultaneously running for not more than 50 and 285 minutes wall clock time, respectively. The main memory consumption when merging sets of contigs decreased by orders of magnitude making the computation feasible for a modern consumer laptop.

Current estimates report that the human genome comprises 50%-65% repetitive sequence with a majority of it being transposable elements [Haubold and Wiehe, 2006; Criscione *et al.*, 2014]. The short-read sequencing technology providing the data for the presented methods as well as the CDBG traversal itself are not beneficial to accurately assemble repetitive sequence. A NRS from a callset of PopIns2 that is predominantly composed of low entropy sequence or short tandem repeats should be treated with caution. Tools like ANISE and BASIL [Holtgrewe *et al.*, 2015] include additional steps during the NRS detection to better resolve the precise order of near-identical repetitive sequences. Another *ad hoc* solution that still detects many reliable and potentially medical relevant NRS is to exclude strongly repetitive NRS [Kehr *et al.*, 2017].

Finally, assigning individuals to continental groups solely by their variants' genotype predictions demonstrates solid predictive power of the genotyping of PopIns2. The credibility of the genotype predictions and the transmission rate of observed alleles closely resemble its predecessor and exceed those of its competing software Pamir. If the number of individuals being processed with PopIns2 is large enough, applying a filter that rigorously selects only variants which comply to the HWE reduces the overabundant number of heterozygous genotype predictions.

Even though the third generation of sequencing technologies is on the rise and its data already has been shown to resolve NRS variants better [Meleshko *et al.*, 2019; Ebert *et al.*, 2021] than NGS, short-read data is still the most prevalent type of sequencing data. With the advance of software and sequencing technology, long-reads will most likely be the unrivalled data of choice for structural variant detection in the near future. Nevertheless, until the cost effectiveness and sequencing error rate of third-generation sequencing technologies will be superior to NGS, accurate and routine NRS detection from short-reads remains highly relevant.

Future Work

The presented work can be extended and complemented by various directions of future research. The suggestions and perspectives provided here can be broadly classified into algorithmic improvements and data analysis.

Long-range connectivity information. An algorithmic approach to consider for an improved genomic sequence assembly is the integration of long-range connectivity information. It was shown [Turner *et al.*, 2018; Jain *et al.*, 2018*a*] that long-range connectivity information can reduce the number of errors in the assembly. Irrespective of the type of assembly (OLC or via DBG), long-range connectivity information can disambiguate the contiguity of sequence fragments during the assembly. Long-range connectivity information exists on different scales. Essentially every continuous sequence or set of related sequences that spans a larger genomic interval than the present fragments of the assembly provides additional information that can guide the assembly. In terms of an assembly that utilizes a de Bruijn Graph these fragments are the k-mers. Therefore, every continuous sequence or set of related sequences longer than k provides additional long-range connectivity information. Considering solely paired NGS reads as original data for the assembly the long-range connectivity information can be either a read itself or the information about the pairing [Turner *et al.*, 2018]. Figure 39 illustrates on a simple case how the pairing of NGS reads solves a local ambiguity during the reconstruction of a genomic sequences from a DBG.

A similar approach was applied to the latest assembly of the haploid CHM13 cell line [Jain *et al.*, 2018*b*, 2020; Nurk *et al.*, 2021]. Here, an accurate assembly from complex local substructures in a *genome assembly graph* was achieved by aligning *ultra-long reads*

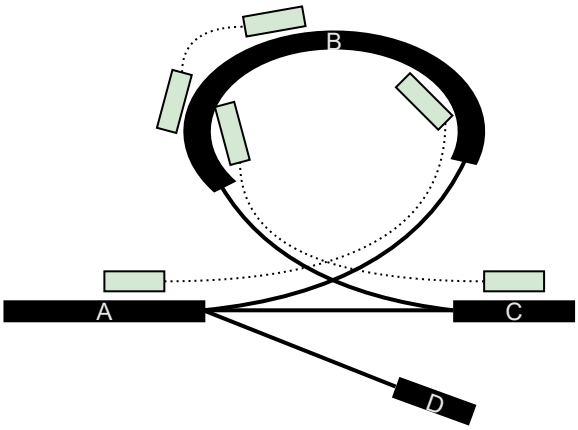

Figure 39: Long-range connectivity information in an excerpt of a compacted de Bruijn Graph. The thin solid lines denote the connections between unitigs A, B, C and D. Without further ado the maximal unitig paths AC, ABC and AD are valid candidates for contig assemblies. However, the read pairs (green) provide long-range connectivity information (thin dotted lines) that guides the traversal through the path ABC.

to the graph. Even though ultra-long reads have a relatively high per base error rate compared to other sequencing technologies they still preserve the long-range structure of genomic sequences that aids solving ambiguities during the assembly [Jain *et al.*, 2018*a*]. The methodology of aligning a sequence to a graph, the structure of assembly graphs and the details about ultra-long reads are beyond the scope of this theses but what is to take away from this approach is that additional reads longer than the original input sequences can guide the assembly in complex graph structures. Mind that in comparison to the read pairs as long-range connectivity information the second approach required additional sequence data.

Regarding future directions of PopIns2 a potential improvement to the merging algorithm could be to include long-range connectivity information into the CDBG, either from read pair information or reads from additional sequencing protocols. The expected outcome of this improvement is more accurate NRS where the merging method had to traverse densely interconnected substructures of the CDBG. Despite the fact that adding long-range connectivity information seems like a promising idea in general it must be monitored whether PopIns2 can still maintain its scalability.

Concurrent graph traversal. Another future project for algorithmic improvement could be a multi-threading mode of the graph traversal. The traversal of the CDBG is trivially parallelizable when applied to individual connected components of the graph. In all presented applications of PopIns2, simulated and real data, the CDBG was composed of many connected components.

Confidence scoring. Thirdly, the individual traversal decisions when generating the paths from the CDBG can be interpreted as following the a heaviest outgoing edge (the edge with the highest Jaccard index) of a vertex. The edge weights of an entire path could be accumulated to a confidence score indicating a certainty for the correctness of the NRS. Also, in addition to the edge weights, the superiority of the heaviest edge to the second heaviest edge can be taken into account similar to the computation[22] of the mapping quality score in [Li, 2013].

Genotyping improvements. Finally, the genotype module can be revised by integrating a more sophisticated genotyping framework. As shown recently [Chen *et al.*, 2019; Eggertsson *et al.*, 2019; Ebler *et al.*, 2020], sequence graphs have great potential to improve the genotyping accuracy for SVs.

Understanding and reducing the reference bias. The other major direction to follow up on the development of PopIns2 is to apply the software to more data sets and diverse reference panels. For instance, a simple but promising adjustment is to apply PopIns2 to sets of reads aligned to the latest Telomere-to-Telomere human genome reference (T2T) [Miga *et al.*, 2020; Nurk *et al.*, 2021]. The authors of the T2T addressed the accurate and comprehensive assembly of a human genome, particularly the estimated 8% genomic sequence (200 million base pairs) previously unidentified in the GRCh38 reference. Utilizing such a more comprehensive reference genome promises to reduce the reference bias [Aganezov *et al.*, 2021]. This approach can be taken even further by applying PopIns2 to only those sequences that do not align to an entire collection of references, e.g. provided by a *pan-genome* graph [The Computational Pan-Genomics Consortium, 2018; Li *et al.*, 2020] or a *r-Index* [Mun *et al.*, 2020].

Comparison to SV callsets from other sequencing technologies. Another data analysis that can shed more light on the extend of NRS that the population-scale approach of PopIns2 is able to detect from NGS data is a comparative study between sequencing technologies on the same data set.

For instance, multiple studies [Sedlazeck *et al.*, 2018; Mahmoud *et al.*, 2019] have shown that long reads strongly improve structural variant calling even though long reads typically come with a higher per base error rate [Jain *et al.*, 2018*a*; Wenger *et al.*, 2019] compared to the shorter reads of NGS protocols and exhibit problems particularly with indels

[22] https://genome.cshlp.org/content/suppl/2008/09/26/gr.078212.108.DC1/maq-supp.pdf

[Amarasinghe *et al.*, 2020; Carneiro *et al.*, 2012; Weirather *et al.*, 2017] and homopolymers [Wenger *et al.*, 2019]. Another sequencing technology that gained a lot of popularity over the last years are so called *linked reads*. With the pluralistic promise to combine the best of each world, the per base accuracy of next-generation short-reads and the long range information of long reads, they rapidly caught the attention of the genomics community and methods [Karaoglanoglu *et al.*, 2020; Elyanow *et al.*, 2018; Fang *et al.*, 2019; Meleshko *et al.*, 2019] for SV detection evolved.

Given a population or study group of individuals sequenced with NGS and at least one other sequencing technology, population-scale SV calling can be applied and the resulting NRS callsets compared. However, even though more and more data sets of many individuals are being generated [Francioli *et al.*, 2014; Beyter *et al.*, 2021] only few release publically available raw data [Zook *et al.*, 2016; Byrska-Bishop *et al.*, 2021; O Connell *et al.*, 2021] and even fewer have data sets from multiple sequencing technologies available. The latter is a promising starting point for additional data analysis comprising publically available NGS and long reads of at least 26 individuals.

Application to other species. Finally, even though this thesis addresses exclusively human genomic sequences the methods can seamlessly be applied to other species.

References

Abel,H.J. *et al.* (2020) Mapping and characterization of structural variation in 17,795 human genomes. *Nature,* **583** (7814), 83–89.

Adams,M.D. *et al.* (2000) The Genome Sequence of Drosophila melanogaster. *Science,* **287** (5461), 2185–2195.

Ade,C. *et al.* (2013) Alu elements: An intrinsic source of human genome instability. *Current opinion in virology,* **3** (6), 639–645.

Aganezov,S. *et al.* (2021). A complete reference genome improves analysis of human genetic variation. Technical report Department of Computer Science, Johns Hopkins University, Baltimore MD, USA.

Alkan,C. *et al.* (2011) Genome structural variation discovery and genotyping. *Nature Reviews Genetics,* **12** (5), 363–376.

Altshuler,D. *et al.* (2005) A haplotype map of the human genome. *Nature,* **437** (7063), 1299–1320.

Amarasinghe,S.L. *et al.* (2020) Opportunities and challenges in long-read sequencing data analysis. *Genome Biology,* **21** (1), 30.

Antonarakis,S.E. *et al.* (2004) Chromosome 21 and down syndrome: from genomics to pathophysiology. *Nature Reviews. Genetics,* **5** (10), 725–738.

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

Azar,Y. *et al.* (1999) Balanced Allocations. *SIAM Journal on Computing,* **29** (1), 180–200.

Bankevich,A. *et al.* (2012) SPAdes: A New Genome Assembly Algorithm and Its Applications to Single-Cell Sequencing. *Journal of Computational Biology,* **19** (5), 455–477.

Becker,T. *et al.* (2018) FusorSV: an algorithm for optimally combining data from multiple structural variation detection methods. *Genome Biology,* **19** (1), 38.

Bellman,R. (1957) *Dynamic Programming.* Dover Publications.

Benjamin,D. *et al.* (2019). Calling Somatic SNVs and Indels with Mutect2. Technical report The Broad Institute, 415 Main Street, 02142 Cambridge, MA, USA.

Beyter,D. *et al.* (2021) Long-read sequencing of 3,622 Icelanders provides insight into the role of structural variants in human diseases and other traits. *Nature Genetics,* **0**, 1–8.

Bhangale,T.R. *et al.* (2005) Comprehensive identification and characterization of diallelic insertion–deletion polymorphisms in 330 human candidate genes. *Human Molecular Genetics,* **14** (1), 59–69.

Bjornsson,E. *et al.* (2021) Lifelong Reduction in LDL (Low-Density Lipoprotein) Cholesterol due to a Gain-of-Function Mutation in LDLR. *Circulation. Genomic and Precision Medicine,* **14** (1), e003029.

Bloom,B.H. (1970) Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM,* **13** (7), 422–426.

Boycott,K.M. *et al.* (2017) International Cooperation to Enable the Diagnosis of All Rare Genetic Diseases. *The American Journal of Human Genetics,* **100** (5), 695–705.

Buniello,A. *et al.* (2019) The NHGRI-EBI GWAS Catalog of published genome-wide association studies, targeted arrays and summary statistics 2019. *Nucleic Acids Research,* **47** (D1), D1005–D1012.

Byrska-Bishop,M. *et al.* (2021) High coverage whole genome sequencing of the expanded 1000 Genomes Project cohort including 602 trios. *bioRxiv,* **0**, 2021.02.06.430068.

Cameron,D.L. *et al.* (2019) Comprehensive evaluation and characterisation of short read general-purpose structural variant calling software. *Nature Communications,* **10** (1), 3240.

Cameron,D.L. *et al.* (2017) GRIDSS: sensitive and specific genomic rearrangement detection using positional de Bruijn graph assembly. *Genome Research,* **27** (12), 2050–2060.

Carneiro,M.O. *et al.* (2012) Pacific biosciences sequencing technology for genotyping and variation discovery in human data. *BMC Genomics,* **13** (1), 375.

Carvalho,C.M.B. and Lupski,J.R. (2016) Mechanisms underlying structural variant formation in genomic disorders. *Nature reviews. Genetics,* **17** (4), 224–238.

Chaisson,M.J. and Pevzner,P.A. (2008) Short read fragment assembly of bacterial genomes. *Genome Research,* **18** (2), 324–330.

Chang,Y.F. *et al.* (2007) The Nonsense-Mediated Decay RNA Surveillance Pathway. *Annual Review of Biochemistry,* **76** (1), 51–74.

Chen,K. *et al.* (2009) BreakDancer: an algorithm for high-resolution mapping of genomic structural variation. *Nature Methods,* **6** (9), 677–681.

Chen,N. (2004) Using RepeatMasker to Identify Repetitive Elements in Genomic Sequences. *Current Protocols in Bioinformatics,* **5** (1), 4.10.1–4.10.14.

Chen,S. *et al.* (2019) Paragraph: a graph-based structural variant genotyper for short-read sequence data. *Genome Biology,* **20** (1), 291.

Chen,X. *et al.* (2016) Manta: rapid detection of structural variants and indels for germline and cancer sequencing applications. *Bioinformatics (Oxford, England),* **32** (8), 1220–1222.

Chikhi,R. and Rizk,G. (2013) Space-efficient and exact de Bruijn graph representation based on a Bloom filter. *Algorithms for Molecular Biology,* **8** (1), 22.

Collins,R.L. *et al.* (2020) A structural variation reference for medical and population genetics. *Nature,* **581** (7809), 444–451.

Compeau,P.E.C. *et al.* (2011) How to apply de Bruijn graphs to genome assembly. *Nature Biotechnology,* **29** (11), 987–991.

Cormen,T.H. *et al.* (2009) *Introduction to Algorithms, Third Edition.* 3rd edition,, The MIT Press.

Crick,F. (1970) Central Dogma of Molecular Biology. *Nature,* **227** (5258), 561–563.

Criscione,S.W. *et al.* (2014) Transcriptional landscape of repetitive elements in normal and cancer human cells. *BMC Genomics,* **15** (1), 583.

Danecek,P. *et al.* (2011) The variant call format and VCFtools. *Bioinformatics,* **27** (15), 2156–2158.

de Cid,R. *et al.* (2009) Deletion of the late cornified envelope LCE3B and LCE3C genes as a susceptibility factor for psoriasis. *Nature Genetics,* **41** (2), 211–215.

Dechering,K.J. *et al.* (1998) Distinct frequency-distributions of homopolymeric DNA tracts in different genomes. *Nucleic Acids Research,* **26** (17), 4056–4062.

Deininger,P. (2011) Alu elements: know the SINEs. *Genome Biology,* **12** (12), 236.

Delage,W.J. *et al.* (2020) Towards a better understanding of the low recall of insertion variants with short-read based variant callers. *BMC Genomics,* **21** (1), 762.

Deorowicz,S. *et al.* (2015) KMC 2: fast and resource-frugal k-mer counting. *Bioinformatics,* **31** (10), 1569–1576.

Dolatabadian,A. *et al.* (2020) Characterization of disease resistance genes in the Brassica napus pangenome reveals significant structural variation. *Plant Biotechnology Journal,* **18** (4), 969–982.

Down,J.L. (1995) Observations on an ethnic classification of idiots. 1866. *Mental Retardation,* **33** (1), 54–56.

Duan,Z. *et al.* (2019) HUPAN: a pan-genome analysis pipeline for human genomes. *Genome Biology,* **20** (1), 149.

Durbin,R. *et al.* (1998) *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, Cambridge.

Earl,D. *et al.* (2011) Assemblathon 1: A competitive assessment of de novo short read assembly methods. *Genome Research,* **21** (12), 2224–2241.

Ebert,P. *et al.* (2021) Haplotype-resolved diverse human genomes and integrated analysis of structural variation. *Science,* **372** (6537).

Ebler,J. *et al.* (2020). Pangenome-based genome inference. Technical report Institute for Medical Biometry and Bioinformatics, Heinrich Heine University Düsseldorf, Düsseldorf, Germany.

Eggertsson,H.P. *et al.* (2019) GraphTyper2 enables population-scale genotyping of structural variation using pangenome graphs. *Nature Communications,* **10** (1), 5402.

Eisfeldt,J. *et al.* (2020) Discovery of Novel Sequences in 1,000 Swedish Genomes. *Molecular Biology and Evolution,* **37** (1), 18–30.

Elyanow,R. *et al.* (2018) Identifying structural variants using linked-read sequencing data. *Bioinformatics,* **34** (2), 353–360.

English,A.C. *et al.* (2015) Assessing structural variation in a personal genome—towards a human reference diploid genome. *BMC Genomics,* **16** (1), 286.

Esrick,E.B. *et al.* (2020) Post-Transcriptional Genetic Silencing of BCL11A to Treat Sickle Cell Disease. *New England Journal of Medicine,* **0**.

Euler,L. (1736) Solutio problematis ad geometriam situs pertinentis. *Commentarii academiae scientiarum Petropolitanae,* **8**, 128–140.

Faber-Hammond,J.J. and Brown,K.H. (2016) Anchored pseudo-de novo assembly of human genomes identifies extensive sequence variation from unmapped sequence reads. *Human Genetics,* **135** (7), 727–740.

Fang,L. *et al.* (2018) NextSV: a meta-caller for structural variants from low-coverage long-read sequencing data. *BMC Bioinformatics,* **19** (1), 180.

Fang,L. *et al.* (2019) LinkedSV for detection of mosaic structural variants from linked-read exome and genome sequencing data. *Nature Communications,* **10** (1), 5585.

Feng,D.F. and Doolittle,R.F. (1987) Progressive sequence alignment as a prerequisitetto correct phylogenetic trees. *Journal of Molecular Evolution,* **25** (4), 351–360.

Fleischmann,R.D. *et al.* (1995) Whole-genome random sequencing and assembly of Haemophilus influenzae Rd. *Science (New York, N.Y.),* **269** (5223), 496–512.

Francioli,L.C. *et al.* (2014) Whole-genome sequence variation, population structure and demographic history of the Dutch population. *Nature Genetics,* **46** (8), 818–825.

Frangoul,H. *et al.* (2020) CRISPR-Cas9 Gene Editing for Sickle Cell Disease and beta-Thalassemia. *New England Journal of Medicine,* **0**.

Fredrick,K. and Ibba,M. (2009) PROTEIN SYNTHESIS. *Nature,* **457** (7226), 157–158.

Fukami,M. *et al.* (2017) Catastrophic cellular events leading to complex chromosomal rearrangements in the germline. *Clinical Genetics,* **91** (5), 653–660.

Gamow,G. (1954) Possible Relation between Deoxyribonucleic Acid and Protein Structures. *Nature,* **173** (4398), 318–318.

Garey,M.R. and Johnson,D.S. (1990) *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., USA.

Garrison,E. and Marth,G. (2012) Haplotype-based variant detection from short-read sequencing. *arXiv:1207.3907 [q-bio],* **0**.

Gaubatz,J.W. (1990) Extrachromosomal circular DNAs and genomic sequence plasticity in eukaryotic cells. *Mutation Research,* **237** (5-6), 271–292.

Goldfeder,R.L. *et al.* (2017) Human Genome Sequencing at the Population Scale: A Primer on High-Throughput DNA Sequencing and Analysis. *American Journal of Epidemiology,* **186** (8), 1000–1009.

Gong,T. *et al.* (2021) Detection of somatic structural variants from short-read next-generation sequencing data. *Briefings in Bioinformatics,* **22** (3), bbaa056.

Gudbjartsson,D.F. *et al.* (2015*a*) Large-scale whole-genome sequencing of the Icelandic population. *Nature Genetics,* **47** (5), 435–444.

Gudbjartsson,D.F. *et al.* (2015*b*) Sequence variants from whole genome sequencing a large group of Icelanders. *Scientific Data,* **2** (1), 150011.

Gurevich,A. *et al.* (2013) QUAST: quality assessment tool for genome assemblies. *Bioinformatics,* **29** (8), 1072–1075.

Hajirasouliha,I. *et al.* (2010) Detection and characterization of novel sequence insertions using paired-end next-generation sequencing. *Bioinformatics,* **26** (10), 1277–1283.

Hardy,G.H. (1908) Mendelian Proportions in a Mixed Population. *Science,* **28** (706), 49–50.

Harvey,W.T. *et al.* (2021) SARS-CoV-2 variants, spike mutations and immune escape. *Nature Reviews Microbiology,* **19** (7), 409–424.

Haubold,B. and Wiehe,T. (2006) How repetitive are genomes? *BMC Bioinformatics,* **7**, 541.

Hehir-Kwa,J.Y. *et al.* (2016) A high-quality human reference panel reveals the complexity and distribution of genomic structural variants. *Nature Communications,* **7** (1), 12989.

Heller,D. and Vingron,M. (2019) SVIM: structural variant identification using mapped long reads. *Bioinformatics,* **35** (17), 2907–2915.

Heller,D. and Vingron,M. (2020) SVIM-asm: structural variant detection from haploid and diploid genome assemblies. *Bioinformatics,* **36** (22-23), 5519–5521.

Hierholzer (1873) Ueber die Möglichkeit, einen Linienzug ohne Wiederholung und ohne Unterbrechung zu umfahren. *Mathematische Annalen,* **6**, 30–32.

Ho,S.S. *et al.* (2020) Structural variation in the sequencing era. *Nature Reviews Genetics,* **21** (3), 171–189.

Holley,G. and Melsted,P. (2020) Bifrost: highly parallel construction and indexing of colored and compacted de Bruijn graphs. *Genome Biology,* **21** (1), 249.

Holtgrewe,M. *et al.* (2015) Methods for the detection and assembly of novel sequence in high-throughput sequencing data. *Bioinformatics,* **31** (12), 1904–1912.

Huang,W. *et al.* (2012) ART: a next-generation sequencing read simulator. *Bioinformatics,* **28** (4), 593–594.

Iafrate,A.J. *et al.* (2004) Detection of large-scale variation in the human genome. *Nature Genetics,* **36** (9), 949–951.

Idury,R.M. and Waterman,M.S. (1995) A new algorithm for DNA sequence assembly. *Journal of Computational Biology: A Journal of Computational Molecular Cell Biology,* **2** (2), 291–306.

International HapMap Consortium (2003) The International HapMap Project. *Nature,* **426** (6968), 789–796.

Iqbal,Z. *et al.* (2012) De novo assembly and genotyping of variants using colored de Bruijn graphs. *Nature Genetics,* **44** (2), 226–232.

Iqbal,Z. *et al.* (2013) High-throughput microbial population genomics using the Cortex variation assembler. *Bioinformatics,* **29** (2), 275–276.

Jaccard,P. (1912) The Distribution of the Flora in the Alpine Zone. *The New Phytologist,* **11** (2), 37–50.

Jain,C. *et al.* (2020). A long read mapping method for highly repetitive reference sequences. Technical report Genome Informatics Section, National Human Genome Research Institute, Bethesda, MD 20892, USA.

Jain,M. *et al.* (2018*a*) Nanopore sequencing and assembly of a human genome with ultra-long reads. *Nature Biotechnology,* **36** (4), 338–345.

Jain,M. *et al.* (2018*b*) Linear assembly of a human centromere on the Y chromosome. *Nature Biotechnology,* **36** (4), 321–323.

Jakubosky,D. *et al.* (2020) Properties of structural variants and short tandem repeats associated with gene expression and complex traits. *Nature Communications,* **11** (1), 2927.

Jeffares,D.C. *et al.* (2017) Transient structural variations have strong effects on quantitative traits and reproductive isolation in fission yeast. *Nature Communications,* **8** (1), 14061.

Jurka,J. *et al.* (2007) Repetitive Sequences in Complex Genomes: Structure and Evolution. *Annual Review of Genomics and Human Genetics,* **8** (1), 241–259.

Jurka,J. *et al.* (2005) Repbase Update, a database of eukaryotic repetitive elements. *Cytogenetic and Genome Research,* **110** (1-4), 462–467.

Jónsson,H. *et al.* (2017) Whole genome characterization of sequence diversity of 15,220 Icelanders. *Scientific Data,* **4** (1), 170115.

Karaoglanoglu,F. *et al.* (2020) VALOR2: characterization of large-scale structural variants using linked-reads. *Genome Biology,* **21** (1), 72.

Kavak,P. *et al.* (2017) Discovery and genotyping of novel sequence insertions in many sequenced individuals. *Bioinformatics,* **33** (14), i161–i169.

Kececioglu,J.D. and Myers,E.W. (1995) Combinatorial algorithms for DNA sequence assembly. *Algorithmica,* **13** (1), 7.

Kehr,B. *et al.* (2017) Diversity in non-repetitive human sequences not found in the reference genome. *Nature Genetics,* **49** (4), 588–593.

Kehr,B. *et al.* (2016) PopIns: population-scale detection of novel sequence insertions. *Bioinformatics,* **32** (7), 961–967.

Kehr,B. *et al.* (2011) STELLAR: fast and exact local alignments. *BMC Bioinformatics,* **12** (9), S15.

Khan,J. *et al.* (2021). Scalable, ultra-fast, and low-memory construction of compacted de Bruijn graphs with Cuttlefish 2. Technical report Computer Science department at the University of Maryland.

Khan,J. and Patro,R. (2021) Cuttlefish: fast, parallel and low-memory compaction of de Bruijn graphs from large-scale genome collections. *Bioinformatics,* **37** (Supplement_1), i177–i186.

Kidd,J.M. *et al.* (2008) Mapping and sequencing of structural variation from eight human genomes. *Nature,* **453** (7191), 56–64.

Kim,H. *et al.* (2020) Extrachromosomal DNA is associated with oncogene amplification and poor outcome across multiple cancers. *Nature Genetics,* **52** (9), 891–897.

Kingsmore,S.F. and Saunders,C.J. (2011) Deep Sequencing of Patient Genomes for Disease Diagnosis: When Will It Become Routine? *Science translational medicine,* **3** (87), 87ps23.

Kirsche,M. *et al.* (2021). Jasmine: Population-scale structural variant comparison and analysis. Technical report Department of Computer Science, Johns Hopkins University, Baltimore MD.

Kosugi,S. *et al.* (2019) Comprehensive evaluation of structural variation detection algorithms for whole genome sequencing. *Genome Biology,* **20** (1), 117.

Krannich,T. *et al.* (2021) Population-scale detection of non-reference sequence variants using colored de Bruijn graphs. *Bioinformatics,* **0** (btab749).

Lander,E.S. *et al.* (2001) Initial sequencing and analysis of the human genome. *Nature,* **409** (6822), 860–921.

Lander,E.S. and Waterman,M.S. (1988) Genomic mapping by fingerprinting random clones: A mathematical analysis. *Genomics,* **2** (3), 231–239.

Lawler,E. (2001) *Combinatorial Optimization: Networks and Matroids*. Dover Publications.

Layer,R.M. *et al.* (2014) LUMPY: a probabilistic framework for structural variant discovery. *Genome Biology,* **15** (6), R84.

Ledford,H. (2020) CRISPR gene therapy shows promise against blood diseases. *Nature,* **588** (7838), 383–383.

Lee,Y.g. *et al.* (2020) Insertion variants missing in the human reference genome are widespread among human populations. *BMC Biology,* **18** (1), 167.

Leffler,E.M. *et al.* (2017) Resistance to malaria through structural variation of red blood cell invasion receptors. *Science,* **356** (6343).

Levy,S. *et al.* (2007) The diploid genome sequence of an individual human. *PLoS biology,* **5** (10), e254.

Li,D. *et al.* (2015) MEGAHIT: an ultra-fast single-node solution for large and complex metagenomics assembly via succinct de Bruijn graph. *Bioinformatics (Oxford, England),* **31** (10), 1674–1676.

Li,H. (2013) Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. *arXiv:1303.3997 [q-bio],* **0**.

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

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

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

Li,H. *et al.* (2020) The design and construction of reference pangenome graphs with minigraph. *Genome Biology,* **21** (1), 265.

Li,S. *et al.* (2013) SOAPindel: Efficient identification of indels from short paired reads. *Genome Research,* **23** (1), 195–200.

Li,Y. *et al.* (2020) Patterns of somatic structural variation in human cancer genomes. *Nature,* **578** (7793), 112–121.

Liu,H.Y. *et al.* (2019) Diagnostic and clinical utility of whole genome sequencing in a cohort of undiagnosed Chinese families with rare diseases. *Scientific Reports,* **9** (1), 19365.

Liu,S. *et al.* (2015) Discovery, genotyping and characterization of structural variation and novel sequence at single nucleotide resolution from de novo genome assemblies on a population scale. *GigaScience,* **4** (s13742-015-0103-4).

Ma,M.J.L. *et al.* (2021) Fetal mitochondrial DNA in maternal plasma in surrogate pregnancies: Detection and topology. *Prenatal Diagnosis,* **41** (3), 368–375.

Magoc,T. and Salzberg,S.L. (2011) FLASH: fast length adjustment of short reads to improve genome assemblies. *Bioinformatics,* **27** (21), 2957–2963.

Mahmoud,M. *et al.* (2019) Structural variant calling: the long and the short of it. *Genome Biology,* **20** (1), 246.

Mallick,S. *et al.* (2016) The Simons Genome Diversity Project: 300 genomes from 142 diverse populations. *Nature,* **538** (7624), 201–206.

Manni,M. and Zdobnov,E. (2020) Microbial contaminants cataloged as novel human sequences in recent human pan-genomes. *bioRxiv,* **0**, 2020.03.16.994376.

Maretty,L. *et al.* (2017) Sequencing and de novo assembly of 150 genomes from Denmark as a population reference. *Nature,* **548** (7665), 87–91.

McColgan,P. and Tabrizi,S.J. (2018) Huntington's disease: a clinical review. *European Journal of Neurology,* **25** (1), 24–34.

McVean,G.A. *et al.* (2012) An integrated map of genetic variation from 1,092 human genomes. *Nature,* **491** (7422), 56–65.

Meddens,S.F.W. *et al.* (2020) Genomic analysis of diet composition finds novel loci and associations with health and lifestyle. *Molecular Psychiatry,* **0**, 1–14.

Medvedev,P. *et al.* (2007) Computability of Models for Sequence Assembly. In *Algorithms in Bioinformatics*, (Giancarlo,R. and Hannenhalli,S., eds), vol. 0, of *Lecture Notes in Computer Science* pp. 289–301 Springer, Berlin, Heidelberg.

Meleshko,D. *et al.* (2019) Detection and assembly of novel sequence insertions using Linked-Read technology. *bioRxiv,* **0**, 551028.

Mendel,G. (1865) Versuche über Pflanzen-Hybriden. *Verhandlungen des naturforschenden Vereines in Brünn,* **Bd.4 (1865-1866)**, 3–47.

Meyerson,M. and Pellman,D. (2011) Cancer Genomes Evolve by Pulverizing Single Chromosomes. *Cell,* **144** (1), 9–10.

Miescher,J. (1871) Ueber die chemische Zusammensetzung der Eiterzellen. In *Medicinsich-chemische Untersuchungen* vol. 4,. Hoppe-Seyler pp. 441–60.

Miga,K.H. *et al.* (2020) Telomere-to-telomere assembly of a complete human X chromosome. *Nature,* **585** (7823), 79–84.

Miller,J.R. *et al.* (2010) Assembly algorithms for next-generation sequencing data. *Genomics,* **95** (6), 315–327.

Mills,R.E. *et al.* (2006) An initial map of insertion and deletion (INDEL) variation in the human genome. *Genome Research,* **16** (9), 1182–1190.

Mills,R.E. *et al.* (2011) Mapping copy number variation by population-scale genome sequencing. *Nature,* **470** (7332), 59–65.

Mohiyuddin,M. *et al.* (2015) MetaSV: an accurate and integrative structural-variant caller for next generation sequencing. *Bioinformatics,* **31** (16), 2741–2744.

Muggli,M.D. *et al.* (2019) Building large updatable colored de Bruijn graphs via merging. *Bioinformatics,* **35** (14), i51–i60.

Muggli,M.D. *et al.* (2017) Succinct colored de Bruijn graphs. *Bioinformatics (Oxford, England),* **33** (20), 3181–3187.

Mullaney,J.M. *et al.* (2010) Small insertions and deletions (INDELs) in human genomes. *Human Molecular Genetics,* **19** (R2), R131–136.

Mun,T. *et al.* (2020) Matching Reads to Many Genomes with the r-Index. *Journal of Computational Biology,* **27** (4), 514–518.

Mäkinen,V. *et al.* (2012) Normalized N50 Assembly Metric using Gap-Restricted Co-Linear Chaining. *BMC Bioinformatics,* **13**, 255.

Mölder,F. *et al.* (2021). Sustainable data analysis with Snakemake. Technical Report 10:33 F1000Research.

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

Nicholson,J.M. and Cimini,D. (2013) Cancer Karyotypes: Survival of the Fittest. *Frontiers in Oncology,* **3**.

Niehus,S. *et al.* (2021) PopDel identifies medium-size deletions simultaneously in tens of thousands of genomes. *Nature Communications,* **12** (1), 730.

Norman,P.J. *et al.* (2017) Sequences of 95 human MHC haplotypes reveal extreme coding variation in genes other than highly polymorphic HLA class I and II. *Genome Research,* **27** (5), 813–823.

Nurk,S. *et al.* (2021). The complete sequence of a human genome. Technical report Genome Informatics Section, Computational and Statistical Genomics Branch, National Human Genome Research Institute, National Institutes of Health, Bethesda, MD USA.

O Connell,J. *et al.* (2021) A population-specific reference panel for improved genotype imputation in African Americans. *Communications Biology,* **4** (1), 1–9.

Ou,S. *et al.* (2020) Effect of sequence depth and length in long-read assembly of the maize inbred NC358. *Nature Communications,* **11** (1), 2288.

Oude Munnink,B.B. *et al.* (2021) The next phase of SARS-CoV-2 surveillance: real-time molecular epidemiology. *Nature Medicine,* **27** (9), 1518–1524.

Pareek,C.S. *et al.* (2011) Sequencing technologies and genome sequencing. *Journal of Applied Genetics,* **52** (4), 413–435.

Payne,A. *et al.* (2019) BulkVis: a graphical viewer for Oxford nanopore bulk FAST5 files. *Bioinformatics,* **35** (13), 2193–2198.

Peng,Y. and Xu,J. (2011) RECOMB. *Lecture Notes in Computer Science,* **5541**, 31–45.

Pevzner,P.A. *et al.* (2001) An Eulerian path approach to DNA fragment assembly. *Proceedings of the National Academy of Sciences,* **98** (17), 9748–9753.

Phillippy,A.M. (2017) New advances in sequence assembly. *Genome Research,* **27** (5), xi–xiii.

Poplin,R. *et al.* (2018) A universal SNP and small-indel variant caller using deep neural networks. *Nature Biotechnology,* **36** (10), 983–987.

Portin,P. (2014) The birth and development of the DNA theory of inheritance: sixty years since the discovery of the structure of DNA. *Journal of Genetics,* **93** (1), 293–302.

Posey,J.E. (2019) Genome sequencing and implications for rare disorders. *Orphanet Journal of Rare Diseases,* **14** (1), 153.

Pritchard,C.C. *et al.* (2016) Inherited DNA-Repair Gene Mutations in Men with Metastatic Prostate Cancer. *The New England Journal of Medicine,* **375** (5), 443–453.

Punnett,R.C. (1926) The Theory of the Gene. *Nature,* **118** (2969), 435–437.

Putze,F. *et al.* (2010) Cache-, hash-, and space-efficient bloom filters. *ACM Journal of Experimental Algorithmics,* **14**, 4:4.4–4:4.18.

Rasmussen,K.R. *et al.* (2006) Efficient q-Gram Filters for Finding All Epsilon Matches over a Given Length. *Journal of Computational Biology,* **13** (2), 296–308.

Rausch,T. *et al.* (2012) DELLY: structural variant discovery by integrated paired-end and split-read analysis. *Bioinformatics (Oxford, England),* **28** (18), i333–i339.

Reinert,K. *et al.* (2017) The SeqAn C++ template library for efficient sequence analysis: A resource for programmers. *Journal of Biotechnology,* **261**, 157–168.

Rizk,G. *et al.* (2014) MindTheGap: integrated detection and assembly of short and long insertions. *Bioinformatics,* **30** (24), 3451–3457.

Rizzi,R. *et al.* (2014) On the complexity of Minimum Path Cover with Subpath Constraints for multi-assembly. *BMC Bioinformatics,* **15** (9), S5.

Roberts,M. *et al.* (2004) Reducing storage requirements for biological sequence comparison. *Bioinformatics,* **20** (18), 3363–3369.

Rédei,G.P. (2008) Protein Synthesis. In *Encyclopedia of Genetics, Genomics, Proteomics and Informatics*. Springer Netherlands Dordrecht pp. 1580–1585.

Sachidanandam,R. *et al.* (2001) A map of human genome sequence variation containing 1.42 million single nucleotide polymorphisms. *Nature,* **409** (6822), 928–933.

Saenger,P. (1996) Turner's Syndrome. *New England Journal of Medicine,* **335** (23), 1749–1754.

Sanger,F. *et al.* (1977) DNA sequencing with chain-terminating inhibitors. *Proceedings of the National Academy of Sciences of the United States of America,* **74** (12), 5463–5467.

Sarma,A. *et al.* (2021) Tracheal aspirate RNA sequencing identifies distinct immunological features of COVID-19 ARDS. *Nature Communications,* **12** (1), 5152.

Schleimer,S. *et al.* (2003) Winnowing: local algorithms for document fingerprinting. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data* SIGMOD '03 pp. 76–85 Association for Computing Machinery, New York, NY, USA.

Schuster,S.C. *et al.* (2010) Complete Khoisan and Bantu genomes from southern Africa. *Nature,* **463** (7283), 943–947.

Schüle,B. *et al.* (2017) Parkinson's disease associated with pure ATXN10 repeat expansion. *npj Parkinson's Disease,* **3** (1), 1–7.

Sebat,J. *et al.* (2004) Large-scale copy number polymorphism in the human genome. *Science (New York, N.Y.),* **305** (5683), 525–528.

Sedlazeck,F.J. *et al.* (2018) Accurate detection of complex structural variations using single-molecule sequencing. *Nature Methods,* **15** (6), 461–468.

Sherman,R.M. *et al.* (2019) Assembly of a pan-genome from deep sequencing of 910 humans of African descent. *Nature Genetics,* **51** (1), 30–35.

Simpson,J.T. *et al.* (2009) ABySS: a parallel assembler for short read sequence data. *Genome Research,* **19** (6), 1117–1123.

Sin,S.T.K. *et al.* (2020) Identification and characterization of extrachromosomal circular DNA in maternal plasma. *Proceedings of the National Academy of Sciences of the United States of America,* **117** (3), 1658–1665.

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

Spielmann,M. *et al.* (2018) Structural variation in the 3D genome. *Nature Reviews Genetics,* **19** (7), 453–467.

Splinter,K. *et al.* (2018) Effect of Genetic Diagnosis on Patients with Previously Undiagnosed Disease. *New England Journal of Medicine,* **379** (22), 2131–2139.

Stankiewicz,P. and Lupski,J.R. (2010) Structural variation in the human genome and its role in disease. *Annual Review of Medicine,* **61**, 437–455.

Sudmant,P.H. *et al.* (2015) An integrated map of structural variation in 2,504 human genomes. *Nature,* **526** (7571), 75–81.

Sutton,G.G. *et al.* (1995) TIGR Assembler: A New Tool for Assembling Large Shotgun Sequencing Projects. *Genome Science and Technology,* **1** (1), 9–19.

Sutton,W.S. (1903) The chromosomes in heredity. *The Biological Bulletin,* **4** (5), 231–250.

Taliun,D. *et al.* (2021) Sequencing of 53,831 diverse genomes from the NHLBI TOPMed Program. *Nature,* **590** (7845), 290–299.

Taylor,J.C. *et al.* (2015) Factors influencing success of clinical genome sequencing across a broad spectrum of disorders. *Nature Genetics,* **47** (7), 717–726.

Telenti,A. *et al.* (2016) Deep sequencing of 10,000 human genomes. *Proceedings of the National Academy of Sciences,* **113** (42), 11901–11906.

The Computational Pan-Genomics Consortium (2018) Computational pan-genomics: status, promises and challenges. *Briefings in Bioinformatics,* **19** (1), 118–135.

Thompson,S.L. and Compton,D.A. (2011) Chromosomes and cancer cells. *Chromosome research : an international journal on the molecular, supramolecular and evolutionary aspects of chromosome biology,* **19** (3), 433–444.

Trapnell,C. *et al.* (2010) Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. *Nature Biotechnology,* **28** (5), 511–515.

Turnbull,C. *et al.* (2018) The 100 000 Genomes Project: bringing whole genome sequencing to the NHS. *BMJ (Clinical research ed.),* **361**, k1687.

Turner,H.H. (1938) A SYNDROME OF INFANTILISM, CONGENITAL WEBBED NECK, AND CUBITUS VALGUS1. *Endocrinology,* **23** (5), 566–574.

Turner,I. *et al.* (2018) Integrating long-range connectivity information into de Bruijn graphs. *Bioinformatics,* **34** (15), 2556–2565.

van Dijk,M.A. *et al.* (2001) Protein sequence signatures support the African clade of mammals. *Proceedings of the National Academy of Sciences of the United States of America,* **98** (1), 188–193.

Vasudevan,A. *et al.* (2020) Single-Chromosomal Gains Can Function as Metastasis Suppressors and Promoters in Colon Cancer. *Developmental Cell,* **52** (4), 413–428.e6.

Venter,J.C. *et al.* (2001) The Sequence of the Human Genome. *Science,* **291** (5507), 1304–1351.

Voelkerding,K.V. *et al.* (2009) Next-generation sequencing: from basic research to diagnostics. *Clinical Chemistry,* **55** (4), 641–658.

Walsh,T. *et al.* (2008) Rare structural variants disrupt multiple genes in neurodevelopmental pathways in schizophrenia. *Science (New York, N.Y.),* **320** (5875), 539–543.

Wang,W.J. *et al.* (2020) Chromosome structural variation in tumorigenesis: mechanisms of formation and carcinogenesis. *Epigenetics & Chromatin,* **13** (1), 49.

Watson,J.D. and Crick,F.H.C. (1953*a*) Genetical Implications of the Structure of Deoxyribonucleic Acid. *Nature,* **171** (4361), 964–967.

Watson,J.D. and Crick,F.H.C. (1953*b*) The Structure of Dna. *Cold Spring Harbor Symposia on Quantitative Biology,* **18**, 123–131.

Weber,J.L. *et al.* (2002) Human diallelic insertion/deletion polymorphisms. *American Journal of Human Genetics,* **71** (4), 854–862.

Weirather,J.L. *et al.* (2017) Comprehensive comparison of Pacific Biosciences and Oxford Nanopore Technologies and their applications to transcriptome analysis. *F1000Research,* **6**, 100.

Wenger,A.M. *et al.* (2019) Accurate circular consensus long-read sequencing improves variant detection and assembly of a human genome. *Nature Biotechnology,* **37** (10), 1155–1162.

Weuve,J. *et al.* (2018) Cognitive aging in black and white Americans: Cognition, cognitive decline, and incidence of Alzheimer disease dementia. *Epidemiology (Cambridge, Mass.),* **29** (1), 151–159.

Wheeler,D.A. *et al.* (2008) The complete genome of an individual by massively parallel DNA sequencing. *Nature,* **452** (7189), 872–876.

White,D. and Rabago-Smith,M. (2011) Genotype–phenotype associations and human eye color. *Journal of Human Genetics,* **56** (1), 5–7.

Wick,R.R. *et al.* (2015) Bandage: interactive visualization of de novo genome assemblies. *Bioinformatics,* **31** (20), 3350–3352.

Wilfert,A.B. *et al.* (2021) Recent ultra-rare inherited variants implicate new autism candidate risk genes. *Nature Genetics,* **53** (8), 1125–1134.

Willems,T. *et al.* (2014) The landscape of human STR variation. *Genome Research,* **24** (11), 1894–1904.

Willson,J. (2020) Resolving the roles of structural variants. *Nature Reviews Genetics,* **21** (9), 507–507.

Wong,K.H.Y. *et al.* (2018) De novo human genome assemblies reveal spectrum of alternative haplotypes in diverse populations. *Nature Communications,* **9** (1), 3040.

Wong,K.H.Y. *et al.* (2020) Towards a reference genome that captures global genetic diversity. *Nature Communications,* **11** (1), 5482.

Xing,Y. *et al.* (2004) The Multiassembly Problem: Reconstructing Multiple Transcript Isoforms From EST Fragment Mixtures. *Genome Research,* **14** (3), 426–441.

Yang,Z. *et al.* (2006) A variant of the HTRA1 gene increases susceptibility to age-related macular degeneration. *Science (New York, N.Y.),* **314** (5801), 992–993.

Yazhini,A. *et al.* (2021) Signatures of conserved and unique molecular features in Afrotheria. *Scientific Reports,* **11** (1), 1011.

Ye,K. *et al.* (2009) Pindel: a pattern growth approach to detect break points of large deletions and medium sized insertions from paired-end short reads. *Bioinformatics,* **25** (21), 2865–2871.

Zarate,S. *et al.* (2020) Parliament2: Accurate structural variant calling at scale. *GigaScience,* **9** (12), giaa145.

Zerbino,D.R. and Birney,E. (2008) Velvet: Algorithms for de novo short read assembly using de Bruijn graphs. *Genome Research,* **18** (5), 821–829.

Zerbino,D.R. *et al.* (2012) Integrating Genomes. *Science,* **336** (6078), 179–182.

Zhao,M. *et al.* (2013) Computational tools for copy number variation (CNV) detection using next-generation sequencing data: features and perspectives. *BMC Bioinformatics,* **14** (11), S1.

Zheng,S. *et al.* (2014) Silent Mutations Make Some Noise. *Cell,* **156** (6), 1129–1131.

Zhou,P. *et al.* (2020) A pneumonia outbreak associated with a new coronavirus of probable bat origin. *Nature,* **579** (7798), 270–273.

Zhu,N. *et al.* (2020) A Novel Coronavirus from Patients with Pneumonia in China, 2019. *New England Journal of Medicine,* **382** (8), 727–733.

Zook,J.M. *et al.* (2016) Extensive sequencing of seven human genomes to characterize benchmark reference materials. *Scientific Data,* **3** (1), 160025.

Appendix

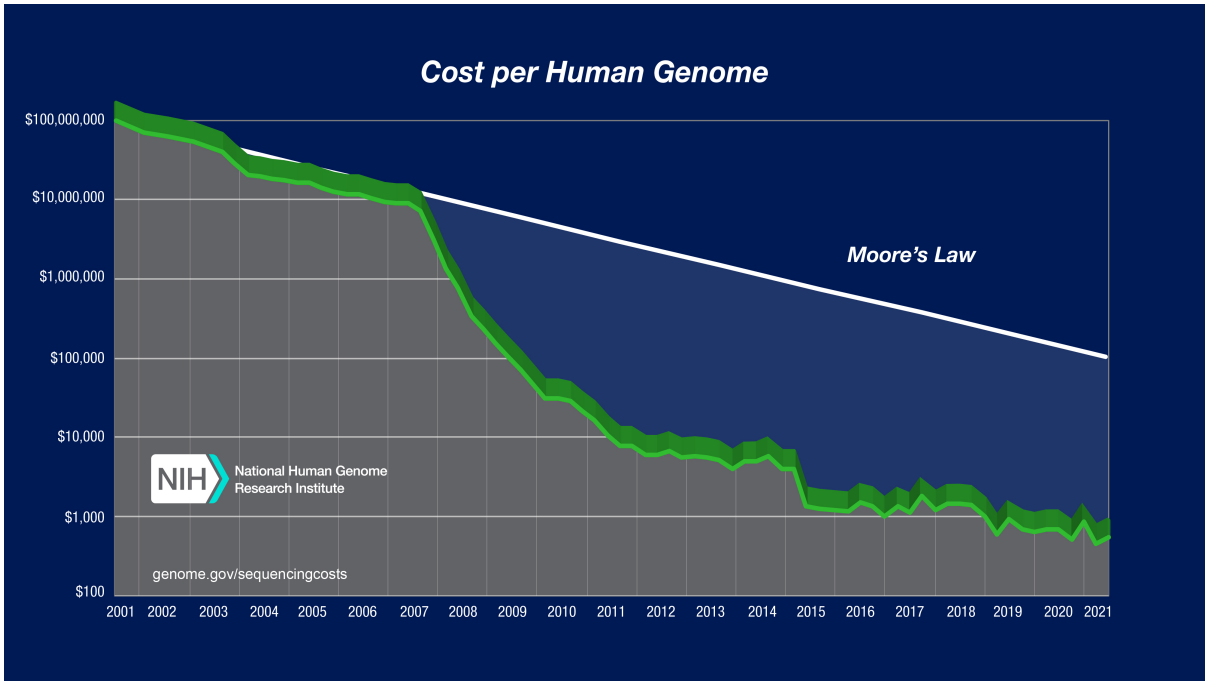

Figure 40: The sequencing cost per genome over the last two decades. With the rise of NGS technologies in 2007, the decrease in cost per individual genome started to outperform *Moore's law*. Moore's law describes the historical trend in the computer hardware industry that the computing power (measured in integrated circuits) doubles roughly every two years. Image was taken from the NHGRI website `https://www.genome.gov/about-genomics/fact-sheets/DNA-Sequencing-Costs-Data`.

Algorithm 1 (Depth First Search in a colored dBG [Krannich *et al.*, 2021])**.** *Given a colored and compacted de Bruijn Graph $G = (V, E, C)$, the Depth First Search (DFS) algorithm searches for the path with the highest consistent set of colors for every startnode $s \in V$. Let u.id be a unique identifier of a unitig $u \in V$ and $G.Adj[u]$ the list of all directly accessible vertexes of u in G. Further, every node maintains a discovery state u.seen determining whether it has been visited by the DFS or not. The set S is a global storage for unitig identifiers. W.l.o.g. the Pseudo-code assumes only one consistent traversal direction. Mind that in a bidirected dBG predecessors and successors have to be determined with respect to the strand of the unitig.*

```
1   DFS-Init(G)
2       for each u ∈ G.V
3           p <- empty path
4           if isStartnode(u)
5               if not DFS-Visit(u, p)
6                   p.append(u)
7               for each u ∈ G.V
8                   u.seen = false
9               if hasEnoughNovelKmers(p, S)
10                  for each q ∈ p
11                      S.add(q)
12                  print(p)
13          if isSingleton(u)
14              p.append(u)
15              if hasEnoughNovelKmers(p, S)
16                  S.add(u)
17                  print(p)
18
19  DFS-Visit(u, p)
20      if u.seen == false
21          u.seen = true
22          if G.Adj[u] is empty:
23              p.append(u)
24              return false
25          G.Adj[u] <- sortByColors(G.Adj[u])
26          for each v ∈ G.Adj[u]
27              if not DFS-Visit(v, p)
28                  p.append(u)
29                  return false
30      return true
```

Selected contributions to other projects during my PhD studies

Bifrost. During the early development phase of the Bifrost library I contributed to the correctness of its functionality by supplying unit tests and bug reports. This contribution has been acknowledged in its original publication [Holley and Melsted, 2020].

Bcmap. Recently, I am involved into a project that focuses on efficient alignment-free barcode mapping of linked-read data. My contribution is the assessment of the approach for a subsequent, local sequence assembly, predominantly for loci of known structural variants. It is intended to submit a manuscript of this work to the ISMB 2022 (Proceeding Submission Deadline on January 13, 2022).

Exploratory workflows for PopIns2. Another recent project I am involved in is the development of workflows for better exploratory experience and automated quality assessment of PopIns2. My contribution is software development and maintenance of a PopIns2 release that is optimized for a better workflow integration.

Index

Selbstständigkeitserklärung

Name: Krannich
Vorname: Thomas Günter Kurt

Ich erkläre gegenüber der Freien Universität Berlin, dass ich die vorliegende Dissertation selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe. Die vorliegende Arbeit ist frei von Plagiaten. Alle Ausführungen, die wörtlich oder inhaltlich aus anderen Schriften entnommen sind, habe ich als solche kenntlich gemacht. Diese Dissertation wurde in gleicher oder ähnlicher Form noch in keinem früheren Promotionsverfahren eingereicht.

Mit einer Prüfung meiner Arbeit durch ein Plagiatsprüfungsprogramm erkläre ich mich einverstanden.

Datum: Unterschrift: