Efficient High Throughput Sequencing Data Compression and Genotyping Methods for Clinical Environments

by

Ibrahim Numanagić

M.Sc., Simon Fraser University, 2013
B.Sc., University of Sarajevo, 2011

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Doctor of Philosophy

in the
School of Computing Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Fall 2016

Approval

Name:	**Ibrahim Numanagić**
Degree:	**Doctor of Philosophy (Computing Science)**
Title:	***Efficient High Throughput Sequencing Data Compression and Genotyping Methods for Clinical Environments***
Examining Committee:	**Chair:** Binay Bhattacharya Professor

S. Cenk Sahinalp
Senior Supervisor
Professor _____

Arrvindh Shriraman
Supervisor
Associate Professor _____

Faraz Hach
Supervisor
Research Associate _____

Leonid Chindelevitch
Internal Examiner
Assistant Professor
School of Computing Science _____

Benjamin J. Raphael
External Examiner
Adjunct Associate Professor
Computer Science
Brown University _____

Date Defended: December 5, 2016

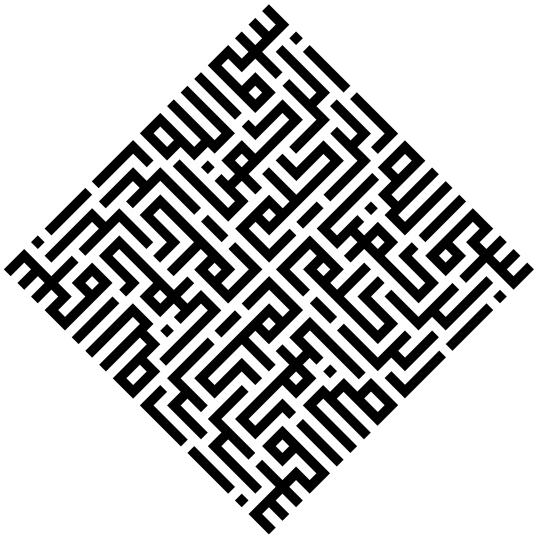
Abstract

The rapid development of high throughput sequencing (HTS) technologies has made a considerable impact on clinical and genomics research. These technologies offer a time-efficient and cost-effective means for genotyping many pharmaceutical genes affecting the drug response (also known as ADMER genes), which makes HTS a good candidate for assisting the drug treatment and dosage decisions. However, challenges like data storage and transfer, as well as accurate genotype inference in the presence of various structural variations, are still preventing the wider integration of HTS platforms in clinical environments. For these reasons, this thesis presents fast and efficient methods for HTS data compression and accurate ADMER genotyping.

First we propose a novel compression technique for reference-aligned HTS data, which utilizes the local assembly technique to assemble the donor genome and eliminate the redundant information about the donor present in the HTS data. Our results show that we can achieve significantly better compression rates over currently used methods, while providing fast compression speeds and random access capability on the compressed archives. We also present a companion benchmarking framework with the aim to evaluate the performance of different HTS compression tools in a fair and reproducible manner.

In the second part, we investigate the genotyping of *CYP2D6* gene. Although this gene is involved in the metabolism of 20–25% of all clinically prescribed drugs, accurate genotype inference of *CYP2D6* presents a significant challenge for various genotyping platforms due to the presence of structural rearrangements within its region. Thus, we introduce the first computational tool which is able to accurately infer a *CYP2D6* genotype from HTS data by formulating such problem as an instance of integer linear programming. Finally, we show how to extend the proposed algorithm to other genes which harbour similar structural rearrangements, like *CYP2A6*, and to other HTS sequencing platforms, like PGRNseq. We demonstrate the accuracy and effectiveness of the proposed algorithms on large set of simulated and real data samples sequenced by both Illumina and PGRNseq platforms.

Keywords: High Throughput Sequencing; Data Compression; Genotyping; *CYP2D6*; *CYP2A6*; PGRNseq



Acknowledgements

First and foremost, let me start by expressing my deepest gratitude to my senior supervisor, Dr. S. Cenk Sahinalp, for his extensive support, guidance and patience during last five years of my studies. I would also like to extend my thanks to Dr. Faraz Hach for all his help and support, and for coming up and leading many projects I have worked on during my PhD studies. I would also like to thank Dr. Arrvindh Shriraman for his valuable comments and suggestions.

This work would not be possible without the people in SFU Computational Biology lab, who were wonderful colleagues during my stay there. I would particularly like to mention all the people with whom I collaborated and whose help was essential in many projects I participated in: Yen-Yi Lin, Phuong Dao, Pınar Kavak, Alexander Gawronski, Ermin Hodžić, Nilgun Donmez and Can Koçkan. And special thanks goes to my colleague Salem Malikić, with whom I worked extensively during my studies, and whose suggestions and proof-reading made this manuscript much more pleasant to read.

Regarding DeeZ and benchmarking projects, I would like to thank James K. Bonfield for his valuable suggestions and comments, and Claudio Alberti for initiating the benchmarking project. I'd also like to mention Jan Voges and Dr. Marco Mattavelli who notably improved the quality of the benchmarking manuscript. As for Cypiripi, I would especially like to thank late Dr. David A. Flockhart, who came up with the project proposal, and also Dr. Victoria M. Pratt and Dr. Todd C. Skaar from Indiana University School of Medicine. Moreover, I would like to acknowledge Dr. Steve Scherer and Dr. Xiang Qin from Baylor College of Medicine, who provided us a large set of real-data samples, and who thoroughly tested Cypiripi.

I would also like to thank Dr. Benjamin J. Raphael, Dr. Leonid Chindelevitch and Dr. Martin Ester for finding the time to examine this work, and for providing valuable remarks which significantly improved the quality of this thesis. Finally, I would like to Vanier Canada Graduate Scholarships for providing me an ample funding during my studies.

Last but not the least, I am thankful to my wonderful family for their unlimited support during my studies. This includes my dear wife Jianqiao Li, my parents Hazim and Šahza, and my siblings Ishak, Kerima and Sadik. That is why this thesis is dedicated to them.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The introduction of high throughput sequencing (HTS) technologies in 2005 [118] made a considerable impact on clinical and genomics research [117]. Compared to the traditionally used Sanger sequencing [158], HTS technologies provide significantly faster operation at lower costs [117, 149], and require a smaller amount of DNA sample needed for sequencing [36]. The first commercially available HTS sequencer, Roche/454 Genome Sequencer [118], offered almost 100-fold increase in the amount of sequencing data produced in a day compared to previous instruments [55]. Later developments, including Illumina Genome Analyzer [13], Illumina HiSeq X, ABI SOLiD [121] and many others [149], kept lowering the cost of sequencing while providing the higher throughput [162, 117]. With the recent introduction of Illumina HiSeq X, the cost of whole human genome sequencing with the coverage of 30× dropped below $1000 mark [182, 80], reaching the target set up by National Human Genome Research Institute (NHGRI) in 2001 [160]. It is expected that the cost of sequencing will decrease even further in the coming years [127].

The sharp drop in sequencing cost made it possible to have HTS machines widely present at university core facilities and even in individual labs [149]. So far, HTS technologies have been successfully used for *de novo* assembly of the underlying genomes [54, 155], genomic and structural variation detection [92, 123], transcriptome characterization [126, 34], cancer analysis [103, 76] and disease discovery [187, 19]. High throughput of HTS machines, combined with their increased robustness, contributed to the launch of large-scale genomic discovery projects operating on large populations [149]. These projects include but not limited to 1000 Genomes Project [3], Genome 10k Project [69], ENCODE Project [41], The Cancer Genome Atlas (TCGA) [171] and Human Microbiome Project [79].

All of the factors mentioned above indicate that HTS technologies have an important role to play in clinical environments, particularly when coupled with large databases obtained from population-wide studies [36]. In clinical settings, HTS can be used to obtain patient's genomic data within a short time (currently it can be done in less than two days [58]). Such sequencing data can be used to obtain clinically important genomic variants that can assist

medical decision making [185, 8, 55]. This makes HTS a great tool for wider adoption of precision medicine, which is an approach to prevent and treat the disease by taking into account the individual variability in genes, environment and lifestyle [144]. The importance of precision medicine was emphasized by Precision Medicine Initiative [26] announced by President of USA [2].

One of the earliest applications of precision medicine was pharmacogenomics, a study of how genetic makeup of individuals affects various drug treatments [7]. Genes involved in the absorption, distribution, metabolism, excretion and response of the drugs are commonly known as ADMER genes. Variations in those genes can affect drug dosing decisions; one such example is the impact of *VKORC1* (which regulates vitamin K essential for the blood clotting) and *CYP2C9* (responsible for metabolizing non-steroidal anti-inflammatory drugs) genotypes on the Warfarin dosage [173]. Other widely known ADMER genes for which the correlation between the genotype and the drug dosage has been established include *CYP2D6* (a Codeine metabolizer [90]) and *CYP2A6* (involved in oxidation of nicotine and cotinine [78]). Moreover, it is currently estimated that the metabolism of 20–25% of clinically prescribed drugs is, at least in part, dependent on *CYP2D6* genotype [81]. Thus, the accurate detection of ADMER genotypes, in particular the *CYP2D6*'s genotype, can significantly impact the treatment decisions.

Several platforms for ADMER gene genotyping have been introduced, including allele-specific primer extension assays, liquid bead arrays and TaqMan genotyping assays [175]. However, there have been reported several discrepancies among genotypes produced by these platforms [143, 45]. Furthermore, discoveries of the novel alleles and variations usually require the extension of existing kits by construction and addition of novel primers, because the existing ones can cause poor or no amplification for novel alleles [48]. Another obstacle for these methods is the presence of various genomic recombinations involving the targeted ADMER gene. For example, both *CYP2D6* and *CYP2A6* are subject to gene duplication and deletion, and in some cases, they form a hybrid gene structure with the evolutionary related pseudogenes *CYP2D7* and *CYP2A7*, respectively [94, 48, 136, 46]. Even the use of Sanger sequencing, notwithstanding its cost and speed, cannot detect the copy number variation or other rearrangements unless that information is known in advance [48].

Many of the challenges described above can be resolved by HTS sequencing and specialized downstream analysis tools [5, 73, 4]. The introduction of cost-effective PGRNseq sequencing platform [59], designed specifically to target the set of 84 functionally diverse ADMER genes, provides further evidence for this claim. In spite of that, integration of HTS technologies into the standard clinical pipeline is not yet straightforward task [36, 7, 33]. Sheer volumes of data generated by a typical HTS sequencing experiment introduces a significant challenge for the underlying computational infrastructure [167, 91]. As an example, typical HTS sequencing experiment with 40× coverage performed on the whole human genome produces approximately 120 GB of raw nucleotide data, without even counting the

accompanying auxiliary information (e.g. base quality scores). The total amount of data generated for a single individual can easily exceed 1 TB [9]. Thus storage, transport and analysis of HTS data pose a major technological challenge, in particular for clinical laboratories [110, 134]. Even if clinical samples are stored and manipulated on cloud, mere transfer and access to the data causes a significant bottleneck [110].

One way of mitigating the storage and data transfer challenges is data compression [75]. Currently, majority of raw HTS data is stored in Gzip or bzip2 file formats [66], which are implementations of popular Lempel-Ziv 77 and Burrows Wheeler Transform schemes [192, 18]. These formats are designed for general purpose data compression, and they are not able to efficiently exploit the inherent properties of sequencing data, such as limited alphabet, genome repeat structure and so on. Specialized tools and formats, tailored specifically for the raw HTS data in the FASTQ file format [140, 25], have shown that better compression rates can be obtained over Gzip or bzip2. Examples include SCALCE [66], DSRC [30, 157], Quip [85], Orcom [60] and Fqzcomp [16].

Another obstacle in the analysis of HTS data is the read alignment. Present-day HTS technologies produce short DNA fragment reads typically ranging from 35 to 300 basepairs (bp) per fragment [149]. The lack of long read fragments is compensated by high genomic coverage (reaching up to 200×) and low error rates (usually less than 1% per base) [80, 35]. Newer technologies, like Pacific Biosciences Single-molecule real-time (SMRT) sequencing [39] or Oxford Nanopore [180], provide significantly longer reads whose length is measured in kilobases. However, these reads are characterized by much higher error rates (15%–25% per base) compared to the currently dominating Illumina technology [149]. Regardless of the sequencer, resulting HTS reads are either used to assemble the original genome, or aligned to the reference genome. *De novo* assembly is computationally intensive problem [125, 22], and it is usually done only in the cases where the reference genome is missing. Read alignment is more lightweight compared to the assembly [7], although still carrying a substantial computational overhead. One issue with the read alignment is the existence of large repeat regions within the genome, which makes the alignment of the short reads originating from these repeat regions ambiguous. In the case of human genome, approximately 5% of the 100bp reads are ambiguously aligned [56]. Various tools have been developed to efficiently deal with the read alignment and to mitigate the issues described above. Well known aligners include Bowtie [100, 99], BWA [106], GEM [116], mrfast [4] and mrsfast [65, 68].

The output of read aligner is usually stored in SAM file format [174], and is further passed to various downstream analysis tools. The SAM file format provides a plain-text human readable file that includes raw reads, as well as the extra information including read's mapping loci, mapping quality and other alignment details [107]. This means that the size of SAM file can easily exceed the size of raw FASTQ file. Thus, efficient SAM compression scheme is desired to lower the burden of data storage. Because the SAM file records are usually sorted by their mapping loci, an important feature to be desired of a

SAM compression scheme is the random access, which allows instant retrieval of the region of interest. For example, if one is to analyze *CYP2D* cluster in the human genome, it would be more practical to access and transfer only the reads spanning the 30 KB-long *CYP2D* cluster than to transfer, decompress and search all 3 GB of the human genome. Current *de facto* SAM compression standard is BAM file format, which is an extension of Gzip file format that supports random access at the expense of larger file size [107].

The last step of HTS data analysis pipeline consists of dedicated downstream analysis tools used for extracting the desired information from aligned data. In the case of ADMER genotyping, variation calling tools, such are Samtools [107] or GATK [120, 32] are used to detect various small nucleotide variants (SNV) in the region of interest and to assess a correct genotype. Many ADMER gene variants are characterized by single nucleotide polymorphisms (SNP) or short insertions/deletions (indels) [166], hence accurate SNV calling is usually enough for proper genotyping. However, ADMER genes prone to significant structural variations (e.g. *HLA*, *CYP2D6* and *CYP2A6*) cannot be accurately genotyped by simple analysis of SNVs [59, 176]. Common reason is that it is not clear whether a read mapping to the gene actually originates from that particular gene, or from the highly similar region in the genome (usually belonging to the adjacent pseudogene). Moreover, popular variant calling tools are not able to detect fusions or similar genomic rearrangements. Finally, the short length of HTS reads further complicates the issue of finding the rearrangement spots in highly homologous regions.

1.1 Contributions

In this thesis, we focus on two previously mentioned computational problems encountered in clinical environments which utilize HTS technologies: (i) reference-aligned data compression and retrieval, and (ii) genotype inference of highly polymorphic ADMER genes. We consider the read alignment problem to be largely addressed in the current literature [64, 100, 4, 106, 116, 65]. By addressing the problems of compression and genotyping, we believe that we can remove some of the major obstacles preventing the efficient integration of HTS technologies into the clinical pipelines which aid the drug prescription decisions tailored for individual patient.

More specifically, we present the following contributions regarding the HTS data compression:

- We introduce DeeZ [67], a SAM/BAM file compression tool which provides significantly improved compression rates over commonly used BAM file format, while providing random access capability to the underlying data. DeeZ achieves this by encoding differences between the donor and reference genome only once. By default, these differences are redundantly encoded in multitude of the read fragments aligning to them. Redundancy elimination is done by implicitly assembling the donor

genome from the available reads. DeeZ also separates various SAM fields in different compression streams, and applies a unique compression method for each stream. Random access is achieved by partitioning the input file into the small blocks. We show that the compression performance of DeeZ matches the performance of state-of-the-art arithmetic coding methods, while providing fast compression times and random access capabilities.

- We present a comprehensive framework for evaluating the performance of various HTS compression tools [131]. This framework comes with a large community-chosen dataset designed specifically to stress test a compression tool on variety of sequencing technologies and species. We have used this framework to evaluate the performance of currently available compression tools, and to gain insight about the kinds of compression techniques that are the most suitable for a particular data type. The proposed framework was developed as a part of the effort initiated by Moving Picture Experts Group (MPEG) to standardize genomic compression formats and methods.

Furthermore, we present fast genotyping methods for genes affecting the drug metabolism which are located in the unstable regions of the genome. Our contributions can be summarized as follows:

- We introduce Cypiripi [132], the first computational tool for exact *CYP2D6* genotype inference by using HTS data with uniform coverage. Cypiripi employs integer linear programming to model and solve the genotyping problem. It is able to properly detect various genomic recombinations, such are gene duplications, deletions and fusions with evolutionary related *CYP2D7* pseudogene. We show that the algorithm performs well on extensive set of simulations designed to cover the majority of known variations and recombinations. Furthermore, Cypiripi's genotype predictions for publicly available CEPH Trio samples match the previously validated calls for those individuals.

- Finally, we provide an extension of Cypiripi, dubbed Cypiripi^{++}, designed to work with PGRNseq and other non-uniform coverage HTS data [59]. Furthermore, Cypiripi^{++} introduces the generalized genotyping model which allows the genotype inference for other ADMER genes. We utilize this model to genotype *CYP2A6* gene, making Cypiripi^{++} the only HTS computational tool which is able to deal with various *CYP2A6* rearrangements. By significantly improving the copy number detection algorithm, Cypiripi^{++} is also the only tool capable of finding the non-functional *CYP2D6*68* allele in CEPH Trio samples. We show that Cypiripi^{++} is able to successfully infer various structural variations and correct genotypes on the large set of real data PGRNseq and Illumina samples.

In addition to the contributions listed above, we also developed (but did not include in this thesis) ORMAN [29], a computational tool which is able to optimally resolve the ambiguously aligned RNA-Seq reads while preserving the novel isoforms.

1.2 Thesis organization

This thesis is divided in two parts. The focus of the first part is HTS data compression, and this part consists of Chapters 2, 3 and 4. In Chapter 2, we describe HTS data file formats, introduce the commonly used compression techniques, and survey the currently available HTS compression tools. Chapter 3 introduces DeeZ, our SAM/BAM file compression tool. Chapter 4 describes our HTS data compression benchmarking framework, and includes the thorough evaluation of the state-of-the-art HTS compression methods.

The second part of the thesis focuses on genotyping of pharmaceutically important genes using HTS sequencing. This part consists of Chapters 5, 6 and 7. In Chapter 5, we first introduce the *CYP2D6* and *CYP2A6* genes, and then proceed with the formal definition of the genotyping problem and a short survey of the currently used genotyping techniques. Chapter 6 describes Cypiripi, a novel method based on Integer Linear Programming which accurately infers the *CYP2D6* genotype. Chapter 7 continues by introducing Cypiripi^{++}, a generalized extension of Cypiripi designed to work with sequencing data of non-uniform coverage and with other ADMER genes, such is *CYP2A6*.

Finally, Chapter 8 provides the summary and conclusion of our contributions, and presents some directions for future work.

Chapter 2

Background on Sequence Compression

The astronomical growth of data generated by the high throughput sequencing (HTS) platforms introduced a major challenge for the computational infrastructure. Data storage, transfer and analysis have become a major obstacle for many labs and clinics worldwide. These challenges have almost signalled the end of Sequence Read Archive (SRA), the world's largest sequence library [91]. Even worse, it is predicted that expected growth of the sequencing data will surpass Moore's law predictions [23]. As an example, current Illumina HiSeq sequencer generates almost a terabyte of data for a single Human sample with 150× coverage. In such circumstances, efficient sequence compression techniques emerge as a major approach for mitigating the storage and transfer challenges.

Although the major sequencing technologies operate in a different fashion, the final result of their software pipelines is a series of short DNA fragments (called *reads*) and their respective quality control values (knows as *quality scores*). In addition, each read might be associated with unique read identifier (or *read name*). This data is presented in FASTQ file format [140, 25], which emerged as a *de facto* standard for raw sequencing data.

After the sequencing process, short read fragments from FASTQ file are usually aligned to the reference genome of the underlying organism, if such genome exists [100, 4, 106, 116, 65]. The result of the alignment is usually a file in a Sequence Alignment/Map (SAM) file format [107, 174] which stores, in addition to the read sequences and associated quality values, alignment information for each read. This alignment information might consist of mapping loci, chromosome identifier, alignment recovery strings and so on. It should be mentioned that in most cases, SAM files are significantly larger than FASTQ files due to the extra overhead generated by the alignment process.

As a final step in the data processing pipeline, SAM files are used as an input to the variant callers or other downstream analysis tools [120, 32, 28, 73]. Results of the downstream analysis tools are usually presented in a Variant Call Format (VCF) file format [28].

Figure 2.1: Sample FASTQ record. Read identifier is "read1".

```
@read1 HS25_09827:2:2109:10656:40243
AAATTGCCTCCAATAGAAACCAGAGTTGCCTGATTACTATCAGCAC
+read1 HS25_09827:2:2109:10656:40243
HGHHGHHGGGHFHGFJHGGJHHIFGGJGIHEGGIFGHGJGJIGGGH
```

Variant call files are usually significantly smaller than their FASTQ or SAM counterparts, and for that reason, we will not deal with the compression of VCF files in this thesis.

In the following sections, we aim to (i) give a quick overview of FASTQ and SAM files, (ii) provide a survey of most important compression algorithms used in sequence compression, and (iii) examine the current state-of-the-art compression tools which operate on HTS data.

2.1 Sequence File Formats

2.1.1 FASTQ

FASTQ is a simple plain-text format describing a set of DNA/RNA read fragments. Over the past years, the FASTQ file format emerged as the *de facto* standard for the storage of unmapped HTS data [140, 25]. Each read block in a FASTQ file consists of four fields (separated by newline characters), containing read identifier (preceded by @ character), nucleotide sequence, read comment (preceded by + character) and read quality score information, respectively. By convention, anything after the first whitespace character in the read identifier field is also considered as a comment, and the string occurring before a whitespace character uniquely identifies the read. A sample FASTQ read block is presented in Figure 2.1.

Quality string Q, consisting of quality values for each nucleotide within the read, is encoded in a Phred format [42]. If the probability of incorrect assignment of the nucleotide b is p, the Phred score $q(p)$ is calculated as

$$q(p) = -10\lfloor \log_{10} p \rfloor.$$

In most observed FASTQ files, comment fields are either empty or a copy of the read identifier fields, and they are rarely (if ever) used; thus, the majority of FASTQ compression tools discards them. Read identifiers generated by the sequencing machine, which usually contain a technical information about the underlying sequencing process (e.g. flow cell identifier or tile information) are also either moved to the comment section or discarded. This is the case for majority of the samples submitted to the Sequence Read Archive.

If the sequencer is able to produce paired-end reads, FASTQ files usually come in the *library* format, where the library consists of two FASTQ files representing the reads from

Figure 2.2: Sample SAM file with two comment lines (preceded by @), and three records.

```
@HD VN:1.5 SO:coordinate
@SQ SN:ref LN:45
r001 163 ref 7 30 8M2I4M1D3M = 37 39 TTAGATAAAGGATACTG *
r002 0 ref 9 30 3S6M1P1I4M * 0 0 AAAAGATAAGGATA *
r003 0 ref 9 30 5S6M * 0 0 GCCTAAGCTAA * SA:Z:ref,29,-,6H5M,17,0;
```

the first and second strand, respectively. In this configuration, identifiers of the k-th read from first file must correspond to the identifiers of the k-th read from the second file within the library.

It is also important to note that the order of the read blocks within a FASTQ file is not important. In the case of paired-end libraries, the only requirement which must hold is that a one-to-one mapping between the reads within the different library files is preserved.

2.1.2 SAM and BAM

The SAM file format [107, 174] is a tab-delimited plain-text representation of the sequences aligned to an arbitrary reference genome. It includes all the data from the FASTQ file, accompanied by additional alignment information for every read, including (but not limited to) mapping position within the reference genome and edit operations necessary to align a sequence to the reference (also known as CIGAR strings). The format is precisely defined in the SAM Format Specification [174]. It suffices to say that information for every read is stored in a single tab-separated line, where each column of the line corresponds to some mapping property (as shown in Figure 2.2). SAM files are typically sorted by their mapping loci, but sorting is not requirement for a valid SAM file.

As can be seen, both FASTQ and SAM files consist of different data fields, such are sequences, qualities, mapping loci and so on. Members of each field share similar properties between themselves, and those properties are very often unique for a particular field. For example, the sequence stream usually consists only of the symbols A, C, T, G and N, while the mapping loci stream usually consists of an increasing sequence of integers. Exploiting the properties of such streams can significantly improve the compression rates. Since those streams are interleaved within SAM or FASTQ files, compressing those files via general purpose tools (such are Gzip, bzip2 and 7-Zip [97, 163, 139]) produces suboptimal results since those tools cannot differentiate between the different fields; they simply treat FASTQ or SAM files as plain-text files. Thus almost every dedicated FASTQ or SAM compression tool handles the different fields separately in order to boost the compression rates.

2.2 General Overview of Compression Strategies

In the most general sense, data compression is "art of reducing the number of bits needed to store or transmit the data" [114]. It can be viewed as a combination of various transformations, probabilistic modelling techniques and encoding strategies [114]. Here, we will present a short overview of techniques commonly used for data compression.

2.2.1 Data transformations

Data transformations usually consist of deduplication techniques, which aim to exploit repetitions within an input stream. Those techniques include run length encoding [156] and the popular dictionary-based Lempel-Ziv schemes [192, 193]. Another widely used technique is the Burrows-Wheeler Transform (BWT) [18], which aims to group together symbols with similar context.

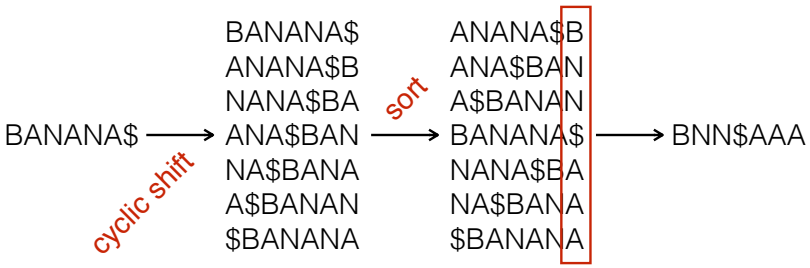
Run length encoding (RLE) [156] is a simple transformation in which a series of consecutive and identical symbols is replaced with the symbol and a number of its occurrences. For example, the string "AAAA" may be encoded as "A4".

Move-to-front encoding (MTF) [14] is a technique based on the premise that the most recent symbol is most likely to occur at a given point in time [114]. It maintains a queue of alphabet symbols (initially ordered alphabetically), and associates to each symbol a rank equal to its position in the queue. At any given time, each symbol from the input is uniquely represented by its rank. After processing the symbol, MTF moves the symbol to the front of the alphabet queue, and updates the ranks accordingly. For example, the string "BBA" is encoded as 101 as follows. The initial queue consists of "AB", so the rank of "B" is initially 1. "B" gets moved to the front of the queue after the encoder encounters it, and its rank becomes 0 while the rank of "A" becomes 1. For data with appropriate local correlations, MFT is able to reduce entropy.

Lempel-Ziv 77 (LZ-77) [192] schemes replace repeated substrings in the input stream with pointers to their previous occurrences. For example, the string "ABRACADABRA" might be encoded as "ABRACAD$(-7, 4)$", where the pointer $(-7, 4)$ has the meaning "step back 7 characters and copy 4 bytes from there". LZ-like schemes keep track of a dictionary consisting of previously seen substrings in order to detect the repetitions. Such a dictionary is often limited to the fixed amount of previously seen characters, which is in practice usually implemented as a fixed "sliding" window of a size between 16 and 32 kB (although some implementations might allow even 4 GB window sizes).

There are various implementations of LZ-77, usually differing from each other in the size of the dictionary and the method used for encoding the pointers. Most commonly used im-

Figure 2.3: An example BWT construction for the string "BANANA". Symbol $ denotes the end of the string, and it is lexicographically the smallest symbol in the alphabet.



plementations are Gzip and zlib [97, 51], which are based on the DEFLATE implementation of LZ-77 [96]. 7-Zip [139] is based on the LZMA variant, while Microsoft's CAB file format is based on LZX [114]. While compression schemes based on LZ-77 vary in compression time and memory usage, decompression is almost universally fast [114].

Due to the abovementioned properties, LZ-77 is ideal for highly repetitive data where similar records are grouped together. This inspired a group of FASTQ compression tools to reorder the reads within a FASTQ file in a way that similar reads get close to each other [138, 60, 66].

Lempel-Ziv-Welch (LZW) [193, 183], a popular extension to the LZ-78 scheme, explicitly constructs a dynamic dictionary, maintained in a simple index structure. Initially the dictionary is comprised of single symbols, each with a codeword equal to the symbol's lexicographic rank in the alphabet. In each iteration of LZW compression, the longest prefix of the uncompressed portion of the input data is identified and replaced by its codeword. This prefix is extended by the next symbol of the input data and the resulting string is inserted in the dictionary with the codeword equal to the number of dictionary entries (after the update). The decompression emulates the steps of compression, extending the dictionary in the same manner in each step.

LZW is the basis of UNIX `compress` program and the GIF image format.

Burrows Wheeler Transform (BWT) [18] is an example of context sorting transformation, in which a characters with similar context are grouped together [114]. This transformation (i) constructs the set of all cyclic shifts (or *rotations*) of the input string s, (ii) sorts them lexicographically, and (iii) outputs the last character of each rotation in a sorted order. The output of step (iii) is called *BWT of a string s*. Example of BWT transformation is shown in Figure 2.3.

Consider BWT transformation of a long English text frequented with the word "the". Sorting the rotations of this text will group rotations starting with the prefix "he" together. Last character of such rotation (and thus the part of BWT) will usually be "t", with a small

chance of possible exceptions (e.g. "brahe" might result in "b" being the part of BWT) [184]. Thus, BWT of such string will contain a long runs of character "t". This property makes BWT an ideal candidate for either RLE or MFT encoding.

A typical example of the tool implementing BWT for the compression purposes is bzip2 [163], which divides the input into blocks of size k (where k is usually 900 KB), and applies BWT with MTF on each block. bzip2 generally provides a better compression than Gzip, but such an improvement comes with a significant speed overhead.

2.2.2 Probabilistic modelling

The problem of compressing sequencing data and metadata is not fundamentally different from any other data compression problem where the statistical properties of the source are not known [114]. A probabilistic model in this context is comprised of probability values associated with observations on the input data, e.g. the observed frequencies of symbols in the input, given preceding combinations of symbols. At any given point, this model can provide a probability distribution for the next symbol in the input data. An entropy encoding scheme can use this distribution to efficiently encode this symbol. The more accurate the model (i.e. predictions) would be, the lower the entropy is and the smaller the encoded output becomes. If, for example, the model predicts the next symbol with probability p, the theoretical encoding requires $\log_2 1/p$ bits [164]. One well known entropy encoding scheme is the (dynamic) Huffman coding [77], which always emits a whole number of bits for each input symbol and thus is optimal only when the symbol probabilities follow "inverse power of two" distribution; the alternative Arithmetic [153] and ANS coding [37] schemes are able to approach the theoretical entropy limit.

Probabilistic models can either be static or dynamic. Static models use the whole input to estimate a probability distribution in advance, and require a separate encoding of such a distribution for decompression. This approach requires to send such a distribution to the decoder as a separate side information. Dynamic models start with a predefined distribution (usually i.i.d.), and keep updating (learning) it with new symbols appearing in the input data stream. They do not require extra information to be stored or transmitted, since both encoder and decoder start with the same distribution and keep "learning" from the input data in the same fashion.

Probabilistic models can also be ranked based on their context and complexity. A model which uses no context to predict the next symbol is considered to be an order-0 model. Such a model typically works with the unconditional probabilities of all input symbols. Higher, order-k models typically use the preceding k symbols to predict the next symbol, effectively building an order-k Markov chain with α^k states, where α is the size of the input alphabet. Higher-order models offer better prediction, but come with higher computational and memory demands, as the number of states grows exponentially with the size of the context.

Extensions to probabilistic modelling include *context mixing* [114] techniques which combine probability distributions of various models into a single mixed distribution. This approach is used by current state-of-the-art compression tools, notably by the PAQ family [115]. Unfortunately, context mining techniques have very high memory usage, and come with substantial computational overhead.

2.2.3 Coding techniques

Unary coding encodes a number n with n occurrences of 1-bit, followed by a single zero bit. For example, 4 is encoded as 11110.

Elias Gamma/Omega Coding [40] is used when the largest possible value in the stream is not known in advance. The Gamma Code represents a number n as a sequence of $\lfloor \log_2 n \rfloor$ zero bytes followed by the binary representation of n. For example, the Gamma Code for 14 is 0001110.

One can iteratively apply Elias Gamma Coding on the value of $\lfloor \log_2 n \rfloor$ to further reduce the number of bits used. The resulting code obtained after $\log_2^* n$ iterations is called Elias Omega Code for the number n.

Delta Coding stores the differences of consecutive elements within the sequence. For example, sequence $1, 2, 3, 5, 6$ can be encoded as $1, 1, 1, 2, 1$ (i.e. $1, 2-1, 3-2, 5-3, 6-5$).

Golomb and Rice Coding [57, 150] are designed for alphabets whose symbol frequencies follow a (two-sided) exponential distribution. Golomb codes use a tunable parameter m to divide the input value n into two parts: $q = \lfloor n/m \rfloor$ and $r = n \bmod m$. The value of q is encoded via unary coding. If $r \leq 2^b - m$, where $b = \lceil \log_2 m \rceil$, r is encoded as binary number with $b-1$ bits; otherwise, $r + 2^b - m$ is encoded as binary number with b bits.

The Rice Code is a restriction of the Golomb code in which the parameter m can only be a power of two. This has the advantage that q and r can be computed using bitwise operators.

Entropy Coders

Entropy coding is used for the input source which has a known statistical model. Here we highlight three examples.

Huffman Coding [77] uses a binary tree in order to generate a *prefix-free* code book for the input symbols. A prefix-free code ensures that no code is prefix of another code in the code book.

The binary tree is constructed as follows. Create a leaf node n_s for each symbol s in the input alphabet Σ, and associate it with a weight w_s (usually correlated to the estimated

probability of symbol s). Insert all newly created nodes into the min-heap. Keep removing two nodes n_{s_1} and n_{s_2} from the heap, and create the parent node m for nodes n_{s_1} and n_{s_2} of weight $w_{s_1} + w_{s_2}$. Add node m to the heap, and continue the procedure until the heap is empty. Finally, each symbol s is encoded as a path from the root of the tree to the leaf node n_s, where right turn in the path is encoded with 1, and left one with 0.

Huffman codes perform best when the probability distribution resembles an exponential distribution with power of 2, since it requires the code lengths to have a whole number of bits.

Arithmetic Coding [153, 152] is a computational technique which uses a given statistical model to achieve an almost optimal encoding of the input. It is able to reach the entropy of the data based on the given model with at most 2 redundant bits [159].

Arithmetic coding assumes that the input symbols are ordered, and that each symbol s has the probability p_s at the given time (which might vary over the time). It also requires the availability of the cumulative probability distribution; i.e. the value of $C(s) = \sum_{s'<s} p_{s'}$ for every s. In its purest form, arithmetic coding keeps shrinking the interval $[a, b)$ (initially $[0, 1)$) into $[a + (b - a) \times C(s), a + (b - a) \times [C(s) + p_s])$. At the end, any number in the final interval represents the arithmetic code of the input.

In practice, the finite precision of modern computers necessitates some adjustments to the original algorithm. Mainly, arithmetic coders have to re-normalize the intervals by rounding their boundaries and multiplying them by some constant c if they become too small. Renormalization often incurs a slight penalty on the compression rate.

Asymmetric Numeral Systems (ANS) [37] are recent alternatives to arithmetic coding with the same theoretical guarantees for the final code lengths [114]. Unlike an arithmetic coder, which maintains two variables a and b for the coding interval, an ANS coder requires only a single variable x to be kept during coding and decoding processes.

ANS maintains two functions, a coder $C : \Sigma \times \mathbf{N} \to \mathbf{N}$ and a decoder $D : \mathbf{N} \to \Sigma \times \mathbf{N}$. The function D must be an inverse of C: $C(D(x)) = x$ must hold for any x. In order to avoid unbounded growth of x, both C and D must maintain the invariant that $x \in I = \{L, L + 1, \ldots, bL - 1\}$ where $L \in \mathbf{N}$ and b is the integer not smaller than 2. If this property is observed, the coder will keep writing x mod b and dividing x with b until x is a member of I. Decoder will follow an analogous inverse process.

It has been shown that as long as the *precursor set* $I_s = \{x \mid C(s, x) \in I\}$ is b-unique (i.e. of the form $k, k + 1, \ldots, bk - 1$), both encoder and decoder will stay synchronized [37]. In practice, values of $L = 2^{23} - 1$ and $b = 2^8$ are commonly used.

Since coding and decoding are inverse operations, they must be executed in the reverse order. In practice, this means that the coder usually encodes an input string from the end

towards the beginning, while decoder outputs the data in the original order. For this reason, dynamic models for ANS coders are not suitable for data streams.

Efficient ANS implementations can utilize look-up tables for significant speed-up, since ANS needs to keep only a single variable for the state. Additionally, ANS allows easy interleaving of multiple ANS streams without any extra overhead [53]. On systems with multiple processor pipelines, interleaving can be exploited to speed up both coding and decoding processes.

Detailed information on the above mentioned and other general lossless compression methods are available through several excellent references in the literature [114, 159].

2.3 FASTQ Compression Tools

FASTQ files are typically compressed with the general purpose tools Gzip and bzip2 [97, 163] tools. Another widely used file format is National Center for Biotechnology Information's (NCBI) SRA [91], which uses the LZ-77 scheme to store the metadata. Unfortunately, more information about it is not available since it is purely designed for internal NCBI use.

Specialized FASTQ compression tools initially apply some sort of data transformation (read identifier tokenization or 2-bit nucleotide encoding) followed by statistical modelling and entropy coding. Examples of such approaches are DSRC and DSRC2 [30, 157], FQC [38], Fqzcomp and Fastqz [16], Slimfastq [43], and LFQC [130].

Because the read order within a FASTQ file is not important, reordering of the reads in a manner which brings the similar reads together can significantly boost the compression rates [66]. This is especially true if the underlying genome is repetitive, or if the coverage of the data is high; in such cases, schemes like LZ-77 can benefit significantly from the improvement of data locality. Tools like SCALCE [66], Orcom [60], Mince [138], and BEETL [27] use this approach as a preprocessing step in order to improve the compression performance.

Another approach is to replace each sequence with a pointer to the underlying reference genome, if such genome is present. LW-FQZip [189] is one such example, and it relies on sequence mapping to obtain a list of pointers. If the reference genome is not available, it can be constructed from the data by assembling the reads into contigs, usually by employing de Bruijn graphs to perform the assembly. Subsequently, a read can be represented as a pointer to an assembled contig, or as a path within a de Bruijn graph. Tools which use assembly for data compression are Quip [85], Leon [12], k-Path [89] and KIC [190]. In general, both sequence assembly and mapping are computationally intensive tasks. However, in order to keep running times reasonable, these tools use specialized versions of these methods which sacrifice biological accuracy for speed, while still providing high compression rates.

Both FASTQ and SAM consist of different data fields, where each field consists of data sharing similar properties (e.g. the sequence field consists only of DNA nucleotides, while mapping loci usually are composed of a non-decreasing set of integers). General

purpose tools, like Gzip and bzip2 [97, 163], treat both SAM or FASTQ as simple plain-text files and produce suboptimal compression rates because they are not able to exploit the underlying data schemata. Simple field decoupling can bring the data with similar properties together, and improve Gzip's (or any other algorithm's) performance significantly, as shown in Table 2.1. For that reason, almost every dedicated FASTQ or SAM compression tool handles different fields separately.

Table 2.1: Effect of field decoupling on Gzip and bzip2's performance with FASTQ files.

Mode	Plain	Separated	Gain
Sample		SRR870667	
pigz	6.94 GB	6.19 GB	752 MB
pbzip2	5.58 GB	5.45 GB	131 MB
Sample		ERR174310	
pigz	18.60 GB	16.19 GB	2,406 MB
pbzip2	14.89 GB	14.12 GB	762 MB
Sample		ERR174324	
pigz	305.69 GB	266.01 GB	40 GB
pbzip2	257.72 GB	229.55 GB	28 GB

Majority of FASTQ tools also use the tokenization scheme for read identifiers, since majority of read identifiers share the common parts. For example, SRA identifiers usually follow the format `SRAsample.1`, `SRAsample.2` etc. Tokenization scheme will separate the sample identifier (in this case, `SRAsample`) from the ordinary number, and encode it only once. After the separation, increasing sequence of ordinary numbers can be efficiently encoded via delta encoding.

2.3.1 General FASTQ tools

All of the tools mentioned in this section retain the ordering of reads within the input FASTQ file.

DSRC and DSRC2 [30, 157] are FASTQ compression tools optimized for industrial use. As such, they boast easy-to-use interface and high throughput of the data. DSRC2 uses advanced read identifier tokenization schemes to boost the read identifier compression rate. Bases are compressed either with a Huffman coder, or with a statistical model of an order up to 9 in conjunction with an arithmetic coder. Quality scores are compressed via either run length encoding, Huffman coding or a statistical model of an order up to 6 which is fed into another arithmetic coder.

FQC [38] aims to provide both lossless and lossy strategy for long time archival of FASTQ files. It encodes the variable parts of read identifiers via delta encoding. Sequences are encoded via 2/8 encoding (i.e. each nucleotide is replaced with 2 bits), while quality scores

are simplified with run length encoding. Finally, processed identifiers and sequences are compressed with LZMA algorithm. Quality scores are compressed with PPM coder [24].

FQC also supports the lossy encoding of FASTQ files. In that case, read identifiers are discarded, and quality scores alphabet is shrank by applying a binning-like scheme, where the similar quality scores are replaced with a single value (e.g. quality scores in the range 33–38 are replaced with the value of 36).

LFQC's [130] key contribution is the advanced read name tokenization scheme. For separate fields, it uses PAQ family of general purpose compression tools [115] to achieve high compression rates. The use of PAQ family as a back engine for compression greatly increases LFQC's CPU overhead.

Fqzcomp and Fastqz [16], originally developed for SequenceSqueeze competition (which they won) [72], use advanced statistical model mixing to achieve very high compression rates. Read identifiers are tokenized and the tokens are encoded with different coders (which consist of either delta or arithmetic coders). Quality scores are encoded by very advanced content mixing model which takes into account the correlations between current and previous symbols. Fzqcomp's model predicts q_i, an i-th quality score within the read, in a context consisting of:

1. q_{i-1},
2. $\max(q_{i-2}, q_{i-3})$,
3. 1 if $q_{i-2} = q_{i-3}$; 0 otherwise,
4. $\min(7, \lfloor \frac{1}{8} \sum_{j=2}^{i} \max(0, q_{j-2} - q_{j-1}) \rfloor)$, and
5. $\min(7, \lfloor \frac{i}{8} \rfloor)$

Sequences are modelled via order-k model, and subsequently encoded via arithmetic coder. For high values of k and high coverage of the data, the encoder can properly learn the underlying reference genome, since context of k nucleotides can uniquely identify the next nucleotide. Fastqz can also make use of reference genome in order to perform the fast hash-based mapping and boost even more the compression rate. Both tools do not support encoding of the read comments.

Slimfastq [43] is a robust industry-oriented reimplementation of Fqzcomp. Compared to Fqzcomp, it provides more stability and support for non-standard FASTQ files.

2.3.2 Reordering tools

Due to the fact that sequencers output the reads in a random fashion, order of the records within a FASTQ file is completely arbitrary. The main idea behind the tools in this section is that much better compression rates can be achieved by grouping the similar reads together. The boosting is caused by the fact that many reads are either the same or highly similar because they come from the same loci or because the genome is highly repetitive.

Note that some alignment software compute statistics on the input FASTQ data with the assumption that this data is randomly distributed, as it is when produced by the sequencing instruments. While it is true that the order is arbitrary and one random order is as meaningless as another random order, removing this "original" randomness can lead to incorrectly computed statistics (e.g. insert size distributions). Therefore, it may be necessary to shuffle any reordered data prior to alignment.

ReCoil [188] construct a similarity graph G between the reads. G is undirected weighted graph with nodes representing the reads. Two nodes are connected with an edge if they share a common k-mer; number of common k-mers represents the weight of an edge. In-memory construction of G is impractical due to the size of typical HTS data set; thus, ReCoil constructs G by externally sorting the list of tuples (k-mer, read ID) based on k-mer content. Then, a list of edges is created by connecting the reads with a common k-mer.

Afterwards, ReCoil calculates the Maximum Spanning Tree (MST) T of G in external memory fashion via Kruskal's algorithm [95], in order to catch the highest similarities between the reads. The final ordering of the reads is obtained by selecting an arbitrary node of T as its root, and traversing the T in a breadth-first fashion. Each read is encoded in differential fashion with respect to its parent read.

SCALCE [66] uses *core substrings* as a measure of similarity in order to group the similar reads together. Those core substrings are generated via Locally Consistent Parsing (LCP). They are further used to cluster the reads into a different bins, where each bin corresponds to the single core substring. Each read within a bin is cyclically shifted based on the position of a core substring, and subsequently each bin is sorted lexicographically based on the shifted reads. Reads are then encoded via 2/8 encoding and compressed via Gzip. Quality scores are encoded via order-3 arithmetic coding model, and read names are just passed to Gzip as-is. SCALCE also supports paired-end libraries, but it does not support compression of the read comments.

Orcom [60] cluster the input reads into the different buckets as follows. For every read, its *minimizer*, defined as lexicographically smallest k-mer within the read, is used as its bucket. The underlying intuition is that two highly similar reads should share the common

minimizers. In order to avoid the uneven distribution of the reads within different buckets, Orcom restricts the definition of minimizers only to those k-mers which do not contain any triplet (AAA, CCC, TTT or GGG) or letter N. After bucketing, reads in each bucket are sorted lexicographically starting from the position of their minimizer. Furthermore, Orcom exploits common overlaps between the reads by encoding such overlaps only once. This is done by maintaining a set M of m previous reads, and trying to match the prefix of the current read with some of the suffixes in M. At the end, nucleotides, overlaps, minimizer positions and other metadata is encoded either via PPM [24] or arithmetic encoder.

Orcom does most of the clustering in-memory in order to minimize the interference of the I/O layer. By doing so, it achieves very high data throughput. Orcom does not support compression of read identifiers and quality scores.

Mince [138] also uses minimizer-based bucketing in order to improve the locality of the reads, but its bucketing criteria is different from the one used in Orcom. Read is assigned to the bucket if the bucket's k-mer is present in the read, and if the intersection of read's k-mers and a set of k-mers formed by the reads already in the bucket is maximal. Since bucketing process is performed in a greedy manner, Mince repeats the bucketing for a reads which have fallen within a very small buckets (usually of size 1) in order to achieve a more balanced bucketing load. Buckets are sorted in the same way as in Orcom, and compressed via Lzip [6], an LZMA-based general purpose compression tool. Mince does not support compression of quality scores, read identifiers and comments.

BEETL [27] uses a generalized version of BWT in order to group the similar nucleotides together. Afterwards, such BWT can be compressed via general purpose compression schemes, such are RLE, LZ-77 or bzip2.

Generalized BWT operates on a set of strings $S = \{s_1, \ldots, s_n\}$ (in the case of HTS data, elements of S represent the reads). Initially, it appends the character $\$_i$ to s_i, where $\$_i$ is always lexicographically smaller than any symbol in the set S. Then, for every s_i, it generates a list of its rotations, and sorts the list of all rotations lexicographically. Last column of such sorted list is the generalized BWT of the set S. This construction requires a large amount of memory for a large set S; for this reason, BEETL does this in external fashion [10].

2.3.3 Alignment tools

Tools in this category aim to align the FASTQ reads to the reference genome, which can be either user-provided or constructed from the reads via assembly. Then, reads can be encoded as simple pointers to the reference genome, with additional list of mismatches if the alignment is not perfect. Most of the mapping or assembly techniques used in this section are designed to perform extremely fast on a large set of reads. The results of such

fast techniques are usually suboptimal from the biological point of view, but good enough for compression purposes.

Quip [85] heavily relies on statistical modelling and arithmetic coding to compress the nucleotides and quality scores. Precisely, nucleotides are modelled with an order-12 Markov chain, while qualities use order-3 statistical model.

One unique feature of Quip is that it can also assemble a small reference genome from the first k reads (by default 1 million) and align subsequent reads to the assembled contigs. By doing so, the whole read can be replaced with the position on the contig (if the read is aligned). Assembly is based on a de Bruijn graphs, where a probabilistic structure called a Bloom filter is used to report the count of each k-mer within a de Bruijn graph G. Bloom filters have a tendency to inflate the count of a particular k-mer (or reporting it as a present, even it if is not), but the probability of such events is low [44].

k-Path [89] construct a de Bruijn graph G of k-mers from the reference, and tries to encode each read as a path within G. The paths within G are encoded via arithmetic coding, where probability distribution of edges depends whether a such edge encodes a substring in the user-provided reference genome or not. For example, if an edge $v \to w$ in G encodes 3-mer ACT, probability of such edge will be much higher if ACT exists in the reference genome. The beginnings of such paths are stored separately in a trie representation and encoded via LZ-77.

k-Path is mainly designed for encoding a RNA-Seq data, since the transcoded part of the reference is much smaller than the whole genomic reference. It does not support compression of read identifiers and quality scores.

Leon [12] uses assembly to construct the reference genome and map reads to it. It does this implicitly by constructing a de Bruijn graph G from the reads, and by encoding each read as a path within G. In order to avoid the excessive memory requirements required for a full-blown de Bruijn graph, Leon uses a Bloom filter to store nodes of G [44], similarly to Quip. Graph edges of the node v in G can be queried via Bloom filters by simply testing all the four successors of the k-mer represented by the node v.

Leon encodes only the starting k-mer for each read, followed by the read's branching information (mapping) within G. All encoded symbols are compressed via order-0 arithmetic coder. Leon supports lossless and lossy quality score compression and uses zlib [51] to compress the qualities. It compresses the read identifiers by applying a differential coding to them, and passes the result to order-0 arithmetic coder.

KIC [190] also constructs a de Bruijn graph G for assembly purposes. It initially counts the abundance of 11-mers within the FASTQ file, sorts the 11-mers by their count, and

constructs the assembly contigs by extending each 11-mer to its neighbouring 11-mer as long as such 11-mer has the count above the threshold t (by default, $t = 3$). The count of visited nodes will be decreased in order to avoid getting stuck in some of the G's cycles. Homopolymers and simple repeats are not used as 11-mers. Reads are then mapped to the assembled contigs and encoded as a position within a contig.

Sequence mappings and other FASTQ fields are subsequently encoded with Gzip or XZ implementation of LZMA scheme [102].

LW-FQZip [189] uses an user-provided reference genome to perform a lightweight mapping of the reads. Reference is indexed by taking note of all k-mers starting with CG. A k-mer within the read which contains the most occurrences in the reference index is selected as read's anchor. Then, a local alignment is performed between a reference and a read around the position of the anchor. For each read, only the mapping data (position and edit operation) is encoded. If read cannot be aligned, it is stored verbatim. Read identifiers are encoded with delta encoding, while quality scores are encoded with run length encoding. Final streams are compressed with LZMA-based compression tools such are Lzip [6] or 7-Zip [139].

2.4 SAM Compression Tools

The current *de facto* standard for SAM files compression is Binary Alignment/Map (BAM) file format. BAM files are stored in a Blocked Gzip Format (BGZF), which is an extension of a Gzip format. In BGZF, every Gzip/DEFLATE block is independent, having no LZ references back to previous blocks. Such encoding produces a slightly lower compression rates compared to Gzip, but allows random access to the records within BAM files. Compressed quality scores represent the major portion of BAM file [107].

BAM file format originated from Samtools [107] suite, which is still the most widely tool used for SAM and BAM processing. In addition to Samtools, few other tools also use BAM as their output format. Notable examples include Picard [17], a Java implementation of BAM standard which ships with optimized Intel Deflater for faster encoding and decoding speeds, and Sambamba [169], which is a heavily parallelized implementation of Samtools for faster encoding and decoding purposes.

All BAM tools support arbitrary ordered SAM files as input, and do not require a reference during the compression or decompression. None of them decouples the different fields during the compression.

Another popular alternative to SAM is the CRAM file format [172], which is a reference-based format that separates different fields and applies a variety of compression techniques on each of them. CRAM is implemented in Cramtools [75], Scramble [15] and recently also Samtools [107] and Picard [17].

In SAM format, reads harbouring the same short nucleotide variation (SNV) are redundantly encoded in an independent manner. Tools like CBC [133] and TSC [178] encode such variations separately in order to decrease the redundancy. Similarly to CRAM, they use a variety of compression techniques on each SAM field.

Finally, both Quip [85] and sam_comp [16] employ highly optimized statistical models for various SAM fields, which puts them among the best performing tools in terms of pure compression ratio.

2.4.1 Reference-based Tools

Cramtools [75] is a reference-based SAM compression tool which decouples the different SAM fields in a separate streams and applies the variety of compression techniques to each stream. For example, mapping loci is encoded with delta coding, and the result of delta coding is stored as a Golomb code, while read and deletion lengths are encoded with Gamma or Huffman codes. Golomb coding is used extensively in the CRAM file format because of its simplicity and speed [133]. Original implementation discarded all quality scores except those for which the associated bases participated in genomic variations. Such quality scores were encoded with Huffman encoding.

CRAM was designed as a replacement for SAM/BAM, and thus supports many use-cases featured in Samtools, such are random access, BAM slicing etc.

Scramble [15] is a C-based implementation of CRAM file format which boasts high parallelism, much lower CPU and memory requirements, and higher compression rates. It keeps quality scores and encodes them with an order-1 statistical model powered by ANS coding. It also tries to estimate on the fly the optimal compression technique for every stream, based on the input data statistics. Unlike Cramtools, Scramble can also compress the reads without the reference, at the cost of lower compression ratio (because reference is generated on the fly and stored with the reads). Many of the Scramble's improvements became part of the new CRAM standard [172].

Statistical modelling tools

Quip [85] also supports compression of the SAM files, and uses the same models as described in the FASTQ section. Additionally, Quip can utilize user-provided reference for higher compression rates. Quip does not provide random access.

sam_comp [16] is an extension of Fqzcomp, and contains the same read identifier, sequence and quality score models for SAM files. It uses context modelling to encode the remaining SAM fields. sam_comp does not support compression of paired-end information and optional fields. sam_comp also does not provide random access

CBC [133] uses statistical modelling for different streams, and encodes them subsequently with the help of either PPM or arithmetic coder. Important feature of CBC is that its model is able to detect common structural variations shared between the reads (e.g. common SNPs or indels), and to exploit the redundancy shared across those variations.

TSC [178] is a modular SAM compression framework, where different modules can be employed for encoding or decoding of different SAM fields or a combinations of fields, respectively. Most of the SAM streams are compressed via zlib. Currently, the focus of the software lies on the low memory compression of the nucleotide sequences. The nucleotide sequence compression algorithm uses a sliding window to exploit the redundancies introduced by the high coverage depth and shared structural variations in a similar fashion as CBC. It uses block-based compression in order to ensure random access to the compressed data.

2.5 Conclusion

In this chapter, we have presented an overview of the existing tools and techniques used for the compression of HTS sequencing data, in particular for the compression of FASTQ and SAM file formats.

FASTQ compression methods can be divided into three groups. The first group relies on various coding schemes in order to encode the reads. Tools from this group offer the most time-efficient way to compress the HTS data, but come with the comparatively lower compression rates. Methods from the second group exploit the similarity between the reads and the reference genome, either by the use of statistical learning, or by explicitly aligning the reads to the reference. This group provides significantly improved compression rates, at the expense of higher computational requirements. Finally, tools from the third group reorder the input reads in order to exploit the similarity between the reads themselves. These tools also have good compression performance, but may require significant amount of memory for read reordering.

SAM compression methods can be categorized based on whether they use a reference genome or not. While methods which do not require a reference offer lower compression rates, they are more convenient for data transfer and archival since they do not require any external dependencies. Another characterization of SAM compression tools can be made based on the utilization of higher-order statistical models. While such models offer the best performance in terms of data compression, their high memory overhead prevents the efficient implementation of random access schemes.

Finally, it should be noted that in the case of both FASTQ and SAM file formats, auxiliary information (such are quality scores and read identifiers) accounts for the major portion of the compressed file, regardless of the underlying compression technique.

Chapter 3

Reference-based Compression by Local Assembly

Recent advances in high throughput sequencing (HTS) technologies caused a burst in generated sequencing data volume [13, 121, 145, 39]. The new Illumina NextSeq platform, for example, generates more than a hundred gigabytes of (uncompressed) sequence data per run for about USD $1000. As HTS data rapidly grows in size, data management and storage have become major logistical obstacles for adopting HTS platforms. As a result, several computational tools [66, 93, 27, 188, 30, 157, 170, 85, 16] have been developed for efficient storage of the raw sequencing data. Unfortunately, the size of sequencing data can grow significantly during downstream analysis, particularly after mapping the reads to the reference genome. The standard approach to store mapped reads is in SAM/BAM [107, 174] file formats. The SAM file format provides a plain-text human readable file that not only includes the raw reads but also information about their mapping loci, mapping quality, mate mapping, etc. Some of these fields are mandatory and some are tool specific. Naturally, such extra information increases the size of the original raw sequencing data, putting additional burden on storage and data transfer. The BAM file format is the Lempel-Ziv (BGZF) compressed version of the SAM file format with some additional information to provide random access. Although BAM file format is offering a better way to store the SAM format, its compression performance is no better than the general purpose Gzip because it does not handle specific field properties separately.

One alternative to the BAM file format is offered by the CRAM file format of the Cramtools and Scramble [75, 15], which aims to provide a better compression ratio. One drawback of Cramtools and Scramble is that they are not lossless: they modify some of the fields of the SAM file format during compression and would not reconstruct them exactly during decompression, which may have effects in further downstream analysis. In addition, Cramtools and Scramble compression (similar to Samtools) does not exploit common features of reads mapped to the same loci that differ from the reference genome, missing some

opportunity for improved compression. As depicted in Figure 3.1, a single nucleotide variant (SNV) in a specific loci supported by multiple reads will be encoded separately for each read by Cramtools. As importantly, Cramtools require significantly higher computational resources than Samtools or many other compression methods.

There are additional compression tools that are based on arithmetic coding (AC) and other data modelling methods, such as Quip [85] and sam_comp [16]. While these tools provide superior compression ratios to Samtools, they again have limited utility as they do not provide random access capability. This can be a major drawback in analyzing large SAM files, as it necessitates the decompression of the whole file (requiring large memory and running time) and manual search of the region of interest.

To address the challenges faced in compression of mapped read data, we present DeeZ, a SAM/BAM file compression tool, which:

1. provides (much) better compression ratio than Samtools, and
2. provides random access capability.

DeeZ's compression performance, which is on par with the state-of-the-art arithmetic coding tools, is a result of its improved redundancy encoding of mapped reads.

The key observation employed by DeeZ is that the vast majority of the nucleotide differences between each read and its mapping locus on the reference genome, is shared with other reads mapped to the same locus. DeeZ aims to lower the cost of representing differences between reads and their mapping locus through "collective" encoding. More specifically, DeeZ obtains the "consensus" of the reads mapped to a specific locus (implicitly "assembling" the donor genome by the use of mapping information provided in a SAM file), and only encodes the differences between the consensus (i.e. implicitly assembled) contigs and the reference genome once. Since there is no difference between the consensus contigs and the reads with the exception of mapping errors or highly allelic regions, DeeZ only encodes the positional information of each read within the relevant contig. Moreover, DeeZ uses a unique compression method for each field of the SAM record in order to exploit its specific properties. For example, read names are tokenized and compressed by the use of delta encoding, while quality scores are (by default) encoded using an order-2 arithmetic coding.

DeeZ provides random access capability by encoding the input SAM/BAM file in a block-by-block manner. Additional features of DeeZ include support for fast flag statistics of a SAM file, and location based read sorting ability (as per Samtools).

DeeZ is available for download at `http://sfu-compbio.github.io/deez`.

3.1 Methods

The key observation used by DeeZ, especially for low error sequencing technologies such as Illumina, is that the vast majority of the nucleotide differences between each read, and the locus on the reference genome it is mapped to, is shared with other reads mapping to the same locus. If our goal is simply to compress the sequence content of a set of reads by encoding the nucleotide differences of each read and the locus it maps to, the only additional information we would need to store is the mapping locus of the read, which, collectively can be compressed very efficiently by the use of run length encoding. As a result, the number of bits used to encode the differences between each read and its mapping loci would dominate the overall encoding.

DeeZ aims to lower the cost of representing differences between reads and their mapping locus through a "collective" encoding. The reads mapped to a particular genome region are locally assembled into contigs and for each read DeeZ only encodes the particular locus of each read within the contig (in case there are some rare read errors or complex allelic differences, additional information is encoded), and encodes the differences between the contig and the reference genome once. Such a saving is especially noticeable on a high-coverage data set, where a single difference (SNV or indel) between the donor genome and the reference will not be redundantly encoded in every read that includes that difference; see Figure 3.1.

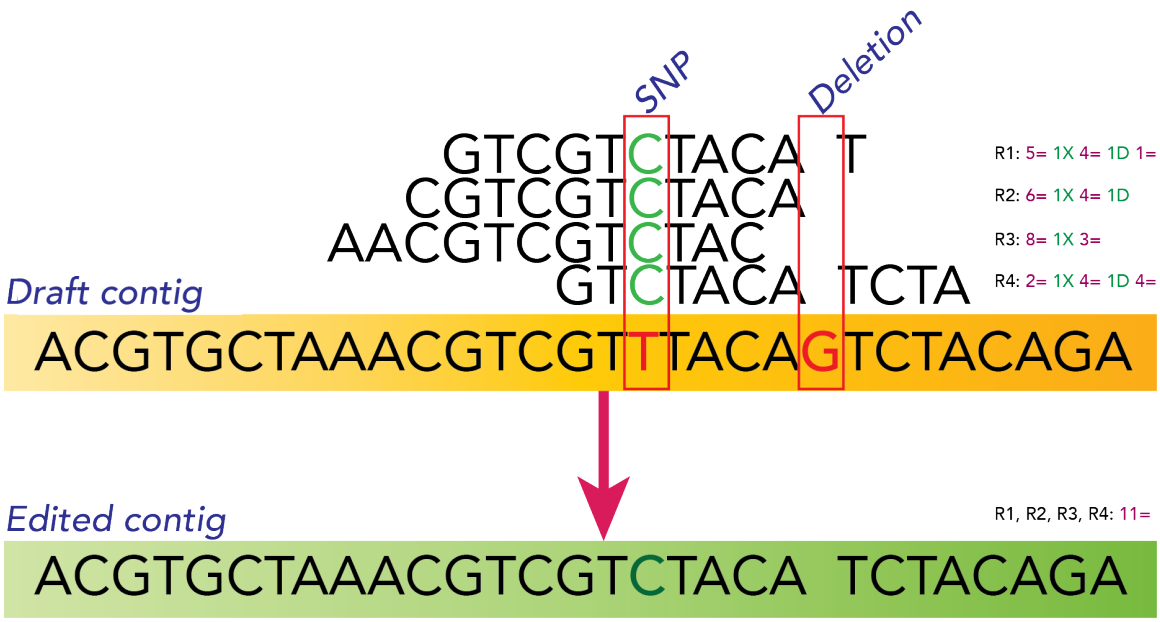
DeeZ represents the donor genome based on the limited assembly of the mapped reads as follows. First, DeeZ partitions the reads into blocks according to their mapping loci, where each block contains a fixed number of reads (the default setting is 1 million—which, on a 40× coverage data set, corresponds to about 25 KB of genome). Then DeeZ processes each block independently from the others and constructs a contig which:

1. covers all of the reads that map to the block, and
2. has the fewest number of edit operations with respect to the reads mapping to the block.

In order to achieve this, DeeZ starts with a draft contig, which satisfies property (1) but not (2)—this contig happens to be the substring of the reference genome which covers all reads mapping to the given block. Then, DeeZ edits the draft contig in a manner that the number of edit operations between the contig and the reads is minimized, as follows.

Given a draft contig Z, we say potential mutation $M_i(x)_Z$ (in the donor genome) substitutes $Z[i]$ with x, where x can represent either a single nucleotide, single nucleotide deletion or insertion. We say that there is "substantial evidence" for $M_i(x)_Z$ if and only if the number of the read mappings supporting $M_i(x)_Z$ is larger than that supporting any other $M_i(y)_Z$ for $y \neq x$ or that supporting no mutation at all. Once DeeZ identifies all potential mutations with substantial evidence, it edits the draft contig Z to include all such

Figure 3.1: DeeZ approach for draft contig editing. Each common mutation supported by the reads is encoded in the resulting contig. The CIGAR strings of mappings are provided on the right side of each read (= represents a match, X a mismatch and D a deletion). After modifying the underlying contig, the originally complex CIGAR strings become simple (only a sequence of = symbols), which boosts the compression significantly.



mutations and obtain contig W. Note that a block deletion can be represented as a sequence of single nucleotide deletions.

Theorem 1. *Given the reference genome and the set of read mappings, a contig W obtained by DeeZ satisfies both properties (i) and (ii).*

Proof. It is trivial to observe that W satisfies (i). To prove that it satisfies (ii), let $f(W)$ denote the number of edit operations necessary to map the reads from the block to contig W. Suppose that there is another contig W', for which $f(W') < f(W)$ and $f(W') \leq f(W'')$ for every other contig. Then, $W' \neq W$ and there should exist a position i on the draft contig Z such that if x_W and x'_W are respectively the nucleotide or indel in W and W' that corresponds to $Z[i]$, then $x_W \neq x'_W$. Obviously $M_i(x_W)_Z$ has more support than $M_i(x'_W)_Z$; otherwise, the DeeZ would have replaced $Z[i]$ with x'_W. But this means that by replacing x'_W with x_W in W', one would end up with a new contig W'' for which $f(W') > f(W'')$, implying a contradiction. □

Thus, by applying the above procedure on the draft contig, DeeZ is able to obtain the optimal contig W which satisfies the above two properties. Since the mapping information and the draft contig are known in advance, this procedure requires linear time and can be performed very fast in practice (unlike typical *de novo* assembly tasks). In addition, DeeZ's block-based design enables one to instantly seek any region in the genome and extract all reads mapping to the region without having to decompress the entire data set.

DeeZ is currently designed to work with both SAM and BAM files. Typical SAM files contain large amounts of additional metadata in addition to the basic read alignment (mapping) information. Such metadata is stored in a different field of the "SAM record", as described in the SAM file reference document [174]. DeeZ groups each field of the SAM file in a separate stream and compresses it independently for each stream. For most fields LZ-77 [192] (Gzip) is used as the compression method of choice since (i) it is fast, and (ii) it has a small overhead as it does not require any *a priori* information about the data set for decompressing a block at any position in the file. In contrast, AC usually needs *a priori* model information (i.e. context) from previously compressed blocks, in order to decompress each given block; such a model needs to be represented for each block independently in order to provide random access capability. Although AC-based tools above perform some kind of implicit assembly themselves, they do it by constructing complex genome models with a large memory footprint and thus are unable to support random access capability.

3.1.1 Reads and CIGAR strings

In the SAM file format CIGAR strings describe the read alignment information needed to correctly map the read to the reference genome (Figure 3.1). DeeZ internally modifies the CIGAR strings to reflect the changes on the edited contigs and to accommodate indels

precisely (since original CIGAR string does not display the insertion details). DeeZ also stores the differences between the resulting contigs and draft contigs (i.e. the reference genome) for accurate reconstruction of the read contents. With these changes, the need of storing reads vanishes completely, since we can reconstruct each read from the resulting contig, its differences from the draft contig and read's corresponding CIGAR string. Note that, because of this encoding scheme DeeZ uses the reference genome for decompression purposes as well. We do not consider this as an obstacle, since only one reference genome is required for whole family of samples from the same species.

DeeZ uses plain LZ-77 to compress unaligned reads, after initially applying 2/8 encoding on them. These reads are not used during the local assembly step, since their mapping locations are unknown.

3.1.2 Read Names

Most sequencers produce unique read identifiers, subsequently referred to as read names, which contain, in addition to the unique read ID, information about the sequencing process and sequencing hardware. This means that many read names share significant amount of information. Thus, we divide each read name into tokens, and compare each token with the token at the same position in the previously processed read name. A new token is encoded only if it differs from the previously processed token. In this way, we are able to significantly decrease the size of read name stream, which is one of the main space consumers in the original SAM file.

3.1.3 Quality Scores

The quality scores are the main obstacle in compression of any next-generation sequencing (NGS) data, due to their higher variability and larger alphabet [179], especially compared to the remainder of a SAM file (for a detailed discussion, please refer to the Results section). Thus, for each quality score string, we first preprocess it by stripping the trailing sequence of usually low quality values at the end of the string, as proposed in [16]. By doing this, we can decrease the length of quality score string while being able to easily reconstruct it during the decompression phase. After that, we pass the string to the simple order-2 arithmetic coder. While more effective schemes for quality score compression exist (e.g. [16]), we found that such schemes require keeping track of various context lengths and build complex models. These context models require high amount of metadata for each block, which are constructed in reference to the previously processed blocks; as a result, in order to decompress the quality score string of a particular block, the entire set of previously processed blocks may need to be decompressed. For users in need of high compression ratios but not random access capability (for quality scores), DeeZ provides the option of using the AC model from sam_comp [16]. In such cases, the users can still seek through

the compressed file, but the quality scores will be not available (although the quality scores can still be obtained by performing full decompression of the compressed file). In addition to this scheme, DeeZ also provides a lossy quality compression scheme as proposed earlier by the SCALCE compression tool [66], which can improve the compression factor by an order of magnitude without any significant impact on a typical downstream analysis.

3.1.4 Mapping Locations

We use delta encoding for compressing mapping locations. In the high coverage samples, delta encoding will represent the mapping location of each read with a small integer (usually either 0 or 1), making the data highly suitable for a further order-0 arithmetic encoding.

3.1.5 Other Features

For all streams, unless otherwise stated, we use the simple Lempel-Ziv 77 (LZ-77) scheme to perform the compression on processed data. This is mainly because our preprocessing and grouping of the SAM fields is designed to increase a locality of reference for each stream, which causes a huge performance boost for LZ-77. In addition, LZ-77 provides both fast compression and decompression with little overhead, which allows fast block seeking through the compressed file. By default, DeeZ will use multiple threads for significantly improving compression and decompression speed (especially the arithmetic coding part, which is an order of magnitude slower than LZ-77 family of algorithms, particularly during decompression).

DeeZ stores the flag statistics of the mapped reads for easy and fast retrieval, and supports reading BAM files as well. Finally, DeeZ supports sorting of the input SAM file with respect to the mapping loci in case the file is unsorted.

3.2 Results

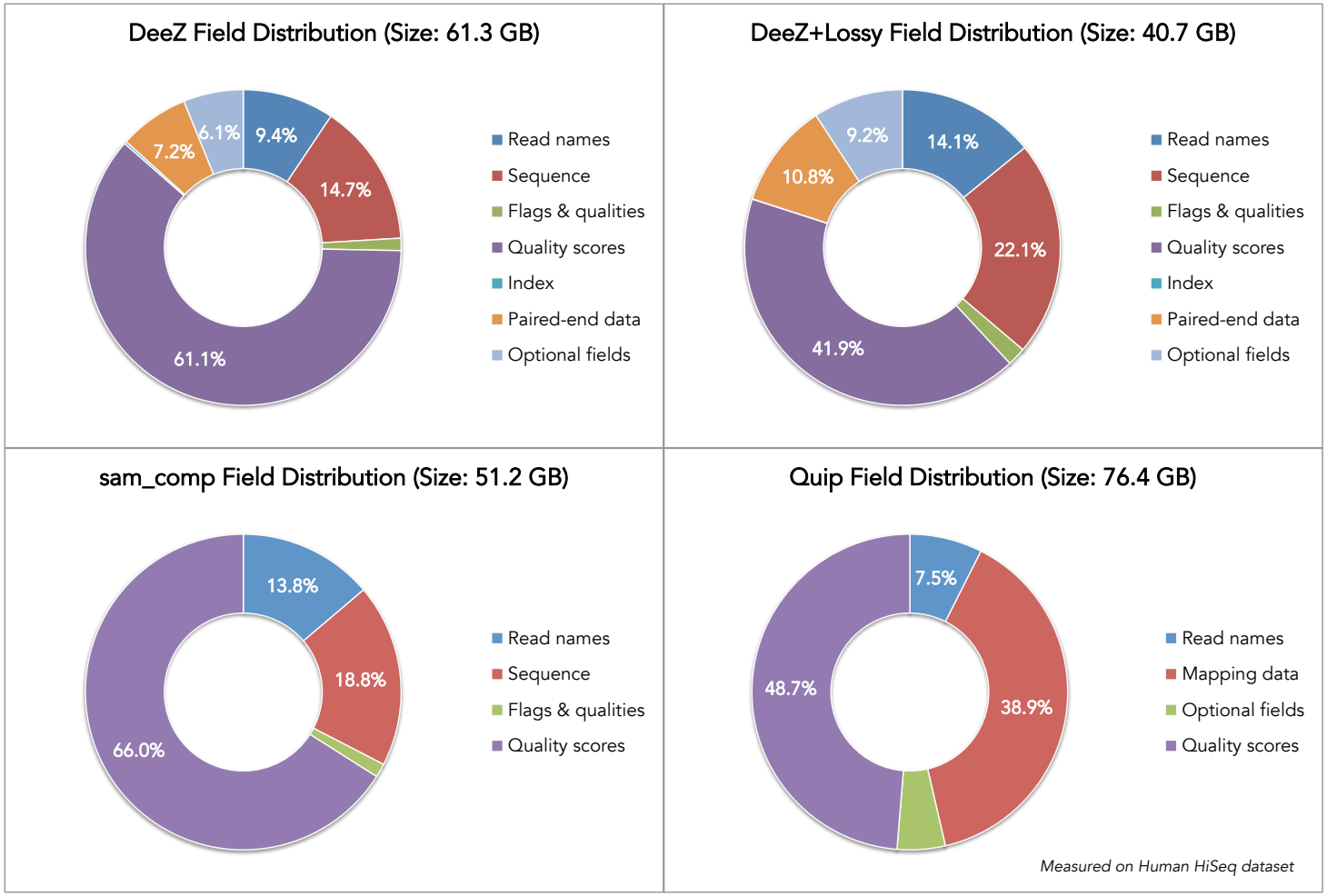
We present how DeeZ compares against other tools on bacterial (*Pseudomonas aeruginosa*) RNA-Seq data as well as Human HiSeq and RNA-Seq libraries.

The following data sets were used for evaluating the performance of DeeZ:

- *Pseudomonas aeruginosa* RNA-Seq library (51bp, sequenced at 700×)
- *E.coli* DH10B MiSeq sample (150bp, file MiSeq_Ecoli_DH10B_110721_PF)
- Human K562_cytosol_LID8465 RNA-seq sample (75bp, accession ID: ERX283488)
- Human NA12878 HiSeq DNA sample (100bp, sequenced at 40×, file NA12878_S1)

P.aeruginosa data set was mapped with BWA 0.7 mapper, in order to produce a valid SAM file. Other data sets were pre-mapped and publicly available as mapped BAM files. All

Figure 3.2: Quality score footprint in compressed files in comparison to other fields (measured on Human HiSeq dataset). Note that different tools internally organize the fields in different manner, thus the difference between chart sections.



data files are valid SAM files with the header and a significant number of unmapped reads (around 1%). The SAM files were sorted by the mapping location coordinate, contained paired-end information, and included several optional fields.

We compared DeeZ with the following compression tools:

1. Gzip v1.3.12
2. Samtools v0.1.19
3. Cramtools v2.0 [75]
4. Scramble v1.13.7
5. Quip v1.1.6
6. sam_comp v0.7 and v0.8
7. Goby v2.3.4

Samtools is the current standard tool for creating and compressing SAM files. Quip is primarily a FASTQ file compression tool with additional support for compressing SAM files. sam_comp is an arithmetic coding based compression tool, not able to compress SAM headers, paired-end information and optional fields. sam_comp and Quip support multiple modes: (i) normal mode, where reads are compressed using arithmetic coding with a specialized context model, and (ii) reference-based mode, where only differences between the reads and the reference are encoded.

Cramtools and Scramble are reference-based compression tools which implement CRAM file format. They are not lossless, i.e. certain fields in the SAM file format are either changed or deleted by Cramtools or Scramble.

Note that in all of the experiments, each tool was run in its default mode, unless otherwise stated. When possible, we chose the options that forces each tool to compress in a lossless fashion as many SAM fields as possible (since some of them discard or modify some fields by default). We provide detailed invocation parameters for each tool in the Appendix A.

Since sam_comp does not support compressing all fields, we compare it only with DeeZ with the option of compressing those fields which sam_comp supports. The compression and timing results are provided in tables 3.1 and 3.2.

Table 3.1: Compression ratios provided by all tested tools. File sizes are reported in megabytes. One megabyte equals 1024×1024 bytes.

			P.aeruginosa RNA-Seq		*E.coli* MiSeq		Human RNA-Seq		Human HiSeq	
Tool	RA	Lossless	Size	Ratio	Size	Ratio	Size	Ratio	Size	Ratio
Original size			19,008	1.00	5,321	1.00	72,398	1.00	437,589	1.00
Gzip	N	Y	3,210	5.92	1,279	4.16	12,236	5.92	99,180	4.41
Samtools	Y	Y	3,340	5.69	1,341	3.97	13,119	5.52	106,596	4.11
Cramtools	N[3]	N	3,967	4.79	N/A	N/A	9,898	7.31	74,564	5.87
Scramble	Y	N	N/A	N/A	1,406	3.79	10,063	7.19	75,784	5.77
Goby	Y	N	N/A	N/A	N/A	N/A	11,757	6.16	N/A	N/A
Quip (non-reference based)	N	Y	2,561	7.42	1,049	5.07	10,601	6.83	78,221	5.59
Quip (reference-based)	N	Y	2,181	8.72	1,135	4.69	8,271	8.75	61,905	7.07
DeeZ	Y	Y	1,921	9.89	831	6.40	8,010	9.04	62,808	6.97
DeeZ (partial random access[2])	P[2]	Y	1,828	10.40	788	6.76	7,615	9.51	58,879	7.43
DeeZ (lossy quality scores)	Y	N	1,343	14.15	513	10.36	5,157	14.04	41,701	10.49
sam_comp[1] (non-reference based)	N	N	1,473	12.91	668	7.96	6,781	10.68	52,389	8.35
sam_comp[1] (reference based)	N	N	N/A	N/A	678	7.85	6,724	10.77	51,733	8.46
DeeZ (sam_comp fields only)	Y	N	1,623	11.71	720	7.39	7,136	10.14	54,435	8.04
DeeZ (sam_comp fields only, partial random access[2])	P	N	1,531	12.41	677	7.86	6,746	10.73	50,536	8.66

[1] sam_comp v0.8 was used to compress *P.aeruginosa* and Human RNA-Seq data set. In other cases, sam_comp v0.7 was used. [2] Quality scores are compressed via sam_comp model and thus are not randomly accessible—the other fields are. [3] Although CRAM file format supports indexing, Cramtools does not provide random access interface. Scramble, C implementation of CRAM specification, does support random access.

Table 3.2: Time and memory usage needed for compression and decompression. All figures are in (H:)MM:SS format. All sizes are in megabytes, unless otherwise specified.

	Compression		Decompression		Compression		Decompression	
	Time	Memory	Time	Memory	Time	Memory	Time	Memory
	P.aeruginosa RNA-Seq				*E.coli* MiSeq			
Gzip	13:35	4	02:15	4	04:44	4	00:45	4
Samtools	14:25	11	02:59	11	04:53	11	00:54	11
Cramtools[1,2]	59:07	>1,120	21:22	>1,120	N/A	N/A	N/A	N/A
Scramble[1]	N/A	N/A	N/A	N/A	06:54	166	02:06	89
Goby[2,5]	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Quip[3] (non-reference based)	14:53	653	16:57	680	05:32	664	05:53	689
Quip[3] (reference based)	14:54	654	17:20	681	05:51	665	05:45	689
DeeZ	12:01	1,350	10:39	1,512	03:49	1,866	04:09	1,880
DeeZ (partial random access)	13:27	1,777	12:20	2,048	04:34	2,285	05:01	2,297
sam_comp[4] (non-reference based)	13:05	474	N/A	N/A	04:45	334	05:35	334
sam_comp[4] (reference based)	N/A	N/A	N/A	N/A	04:43	338	05:24	338
	Human RNA-Seq				Human HiSeq			
Gzip	0:47:24	4	08:35	4	7:31:42	4	2:12:10	4
Samtools	0:53:13	11	10:59	11	6:49:42	11	2:11:33	11
Cramtools[1,2]	2:30:29	>1,220	52:02	>1,220	14:18:38	>1,330	5:28:58	>1,330
Scramble[1]	0:36:04	1,997	11:28	1,989	6:36:41	417	1:59:39	162
Goby[2,5]	5:02:57	>8,600	5:25:09	>8,600	> 1 day	N/A	N/A	N/A
Quip[3] (non-reference based)	1:00:31	846	57:30	870	6:45:12	731	7:24:07	748
Quip[3] (reference based)	0:56:10	1,918	57:04	1,942	8:34:38	1,805	7:27:23	1,822
DeeZ	1:18:10	2,202	48:42	2,074	5:49:48	2,129	6:34:26	2,615
DeeZ (partial random access)	1:27:22	2,681	54:51	2,579	6:50:33	2,532	7:42:04	3,216
sam_comp[4] (non-reference based)	0:51:12	474	56:30	474	6:03:46	334	N/A	N/A
sam_comp[4] (reference based)	0:50:29	717	56:54	717	6:00:33	576	N/A	N/A

[1] Cramtools and Scramble decompressed SAM file was missing 1 GB in the first dataset, 4 GB in the third and 17 GB in the fourth data set. [2] Cramtools and Goby are written in Java, and thus the virtual memory usage is heavily affected by the Java runtime (JRE). On our test machine, JRE was using around > 10 GB of virtual memory. Thus, in those cases we opted to report residential memory usage, which, although being less accurate than the virtual memory usage, provides better insight in the memory usage of Java tools. [3] Quip decompressed SAM file was missing 1 GB in the fourth data set. [4] sam_comp v0.7 was able to decompress only *E.coli* dataset. sam_comp v0.8 succeeded in decompressing the Human RNA-Seq dataset. [5] Goby successfully compressed only RNA-Seq dataset. HiSeq dataset compression took more than one day, and thus we decided to omit its results due to the time constraints.

As it can be seen from the tables, DeeZ outperforms all of the above mentioned tools, with exception of sam_comp, whose compression performance is comparable to DeeZ. However not only does sam_comp not provide random access ability but it also does not compress all fields in the SAM file format (see Table 3.3 for random access performance of DeeZ in comparison to the only other two methods that provide this capability: Samtools and Scramble). In addition, we were not able to decompress any of the files compressed with sam_comp (with the exception of *E.coli* MiSeq dataset). Note that in the HiSeq data set Quip performs slightly better than DeeZ with the default settings due to its use of high-order AC compression for quality scores, which is well suited for a large file. However, Quip does not provide random access ability either.

For users in need of high compression ratios but not random access capability (for quality scores), DeeZ provides the option of using the AC model from sam_comp [16]. With this quality model, DeeZ outperforms Quip on this data set, while still providing partial random access ability (all fields except quality scores).

DeeZ also provides the fastest compression speed in the bacterial RNA-Seq and the human HiSeq data sets. In the human RNA-Seq data set, DeeZ compression speed is lower due to properties of RNA-Seq mapping on eukaryotic genomes; this is due to many mappings being located on exon/intron junctions. Reads mapping to junctions cause DeeZ spend more time analyzing and editing draft contigs[1].

DeeZ's decompression speed is also on par with or better than its competitors with the exception of Samtools and Scramble. This is due to the LZ-77 decompression scheme employed by Samtools and Scramble being much faster than AC decompression, which DeeZ employs for compressing the quality scores. This issue is especially acute in the whole chromosome retrieval task in Table 3.3, where decompression of quality scores dominates the time for random access. In case the quality scores are not needed for an fast random access task, the performance of DeeZ gets improved.

3.2.1 Quality scores

Quality scores usually consume the largest portion of a compressed file, due to their high entropy compared to other fields of a SAM file. Figure 3.2 depicts the size distribution of various SAM fields for Quip, sam_comp and DeeZ in their default settings; as can be seen, even with powerful AC methods, quality scores typically occupy more than half of the file size. As a result, DeeZ provides an optional lossy quality transformation as described earlier [66]. In this way, DeeZ is able to significantly decrease the compressed file size in its default mode (without even using an advanced model), as well as the portion of the file occupied by the quality scores without significantly impacting standard downstream analyses.

[1] Obviously the running times may vary due to I/O utilization and caching. Up to 25% variation between two runs of the same method, on the same machine and same data set is possible.

Table 3.3: Random access performance for three tools which support it. All figures are in (MM:)SS format. Second table indicates the index size (in megabytes) and additional preprocessing (i.e. index building) times needed for Samtools and Scramble.

	Human RNA-Seq		Human HiSeq	
	Time	# records	Time	# records
	chr5			
Samtools	60	9,116,311	08:26	71,141,857
Scramble	63	9,116,311	08:19	71,141,857
DeeZ	119	9,116,311	16:30	71,141,857
DeeZ (without qualities)	84	9,116,311	13:54	71,141,857
	chrY:10,000–20,000			
Samtools	1	26	1	759
Scramble	1	26	3	759
DeeZ	5	26	14	759
DeeZ (without qualities)	3	26	7	759
	chr14:107,349,000–107,349,540			
Samtools	1	0	1	0
Scramble	5	0	1	0
DeeZ	1	0	3	0
DeeZ (without qualities)	1	0	3	0

(a) Random access performance

	Human RNA-Seq		Human HiSeq	
	Time	Size	Time	Size
Samtools	04:04	5.51	30:10	8.63
Scramble	02:55	0.39	05:10	2.03

(b) Index statistics

3.3 Conclusion

In this chapter, we have presented a novel compression tool, named DeeZ, which uses local assembly to improve the compression performance of SAM/BAM files. DeeZ offers high compression rates (up to 50% over commonly used BAM file format), while requiring low computational resources and providing features such as random access. As the rate of HTS data increase surpasses the Moore's law predictions [23], DeeZ can reduce the burden of data storage and transfer by efficient compression and representation of aligned HTS data.

There are still some challenges that we aim to address in future. One of them is to incorporate more complex events into the assembly procedure. In its current iteration, DeeZ only considers single nucleotide variations during the local assembly stage in order to avoid the high computational overhead. Further investigation is needed to ascertain whether the detection of insertions, large deletions and other structural variations significantly improves the compression. Another important concern to be considered is the compression of long reads with high error rates (e.g. reads generated by PacBio or Oxford Nanopore technologies). DeeZ is currently optimized for technologies with low error rates. Thus its performance on data sets where error rates are high might be suboptimal. Furthermore, reordering schemes like those proposed in SCALCE [66] can be utilized for further improving the compression of unaligned reads.

Chapter 4

Comparison of High Throughput Sequencing Data Compression Tools

Current trends in high throughput sequencing (HTS) data generation indicate that the storage, transmission and I/O bandwidth costs will soon surpass the costs of sequencing and will become the main bottleneck in genomics as well as its applications to precision medicine. One way to reduce the burden of HTS data on storage, I/O bandwidth and transmission is the use of high-performance compression methods, developed specifically for HTS data.

In the last 25 years the Moving Picture Experts Group (MPEG), also known as the ISO/IEC JTC1/SC29/WG11 committee, has developed a methodology that has yielded compression standards extensively adopted by the digital media industry. A growing number of experts in genome data processing have joined MPEG experts in a working group, to explore how data compression expertise from the multimedia world can integrate specific bioinformatics expertise and improve the performance of existing genomic data compression tools. In addition, ISO Technical Committee 276 (Biotechnology) Working Group 5 (Data Processing and Integration) has joined the effort with its specific biotechnology expertise. The ultimate goal of this working group is to design and specify genomic data compression and transport technology by means of an open standard and interoperability among systems.

As a first step towards developing an open standard, MPEG and ISO TC 276 has now issued a call to the international community to jointly evaluate the effectiveness of available compression methods on a common set of genome data. For this purpose, the MPEG HTS compression working group compiled a HTS data set with a wide spectrum of characteristics for ensuring statistically meaningful results: raw (FASTQ) and aligned (SAM/BAM) data with both deep and shallow coverage; fixed length and variable length reads obtained by sequencing technologies from leading manufacturers (Illumina, Pacific Biosciences, Oxford

Nanopore, Ion Torrent); genome, exome and transcriptome data from various organisms (*Homo Sapiens*, Bacteria, Plants, Insects); several sample types (metagenomic, cancer cell lines) as well as simulated human data are included in the final dataset of size of 4 TB. The dataset was reviewed and approved by all members of the MPEG HTS compression working group for benchmarking purposes. It is expected that the dataset will grow further to accommodate future technologies and additional requirements.

As members of the MPEG HTS compression working group, we have conducted a comparative study of all available lossless HTS compression tools on the MPEG benchmarking data set, expanding significantly some of the recent comparative studies and surveys [52, 72, 31]. We developed an open-source, publicly available framework specifically tailored for HTS compression evaluation, placing a special emphasis on fairness and reproducibility of the benchmarking process (https://github.com/sfu-compbio/compression-benchmark). Together with the data diversity provided by the MPEG benchmarking data set, this framework is also suitable for the review and comparison of future tools.

4.1 Summary of Available Tools

We aimed to evaluate all available approaches used for HTS data compression. These include both industry-scale tools as well as research-oriented prototypes. For each tool, its compression performance, running times, memory usage, and parallelization capabilities were measured. Most HTS data is maintained either as raw sequencing information in a FASTQ file, or as reference-aligned (mapped) data in SAM or BAM file formats. The FASTQ and SAM schemata describe different data fields with similar properties (e.g. the sequence field consists only of DNA nucleotides, while mapping loci are usually represented as a non-decreasing sequence of integers), generating large files. It is common to use general purpose compression tools on these files, which treat them as simple plain-text and thus produce suboptimal compression rates because they are not able to exploit the underlying data schemata. All sequencing compression tools are built on a standard set of general compression algorithms, which are surveyed in the Chapter 2.

FASTQ files are typically compressed with general purpose Gzip and bzip2 tools. Sequence archives commonly use NCBI's SRA file format, which is loosely based upon the LZ-77 scheme. Specialized FASTQ compression tools initially perform a form of transformation (read identifier tokenization or 2-bit nucleotide encoding) followed by statistical modelling and entropy coding. Examples of such approaches are DSRC2 [157], FQC [38], Fqzcomp and Fastqz [16], Slimfastq [43] and LFQC [130].

Because the read order within a FASTQ file is arbitrary, reordering the reads in a manner which brings the similar reads together can significantly boost the compression rates [66]. This is especially true if the underlying genome is repetitive, or if the coverage of the data is high; in such cases, schemes like LZ-77 can benefit significantly from the improvement of

data locality. Tools like SCALCE [66], Orcom [60], Mince [138], and BEETL [27] use this approach as a preprocessing step in order to improve the compression performance.

Alternative approaches aim to achieve compression by replacing each read with a pointer to the underlying reference genome, provided such a reference genome is available. LW-FQZip [189] is one such example which relies on sequence mapping to obtain a list of pointers. If the reference genome is not available, it can be constructed *de novo* by assembling the reads into contigs, usually through the use of de Bruijn graphs. Subsequently, a read can be represented as a pointer to an assembled contig, or as a path within a de Bruijn graph. Primary tools that use assembly for data compression are Quip [85], Leon [12], k-Path [89] and KIC [190]. Note that both sequence mapping and assembly are computationally intensive tasks; as a result, most of the above mentioned tools sacrifice speed for maintaining high compression rates.

SAM files are mostly stored in their compressed equivalent, the BAM file format [174]. Commonly used tools for BAM manipulation are Samtools [107], Picard [17], and Sambamba [169]. All BAM-based tools support arbitrary ordering of the reads and do not require a reference during compression or decompression. None of them treat various streams in a BAM file differently.

An alternative to the SAM file format is CRAM [172], a reference-based format that separates different fields in the reads and applies a variety of compression techniques on each. CRAM is implemented in Cramtools [75], Scramble [15] and recently also Samtools [107] and Picard [17].

In both SAM and CRAM, reads harbouring the same sequence variant are encoded independently. This implies that the same variant is redundantly encoded across the reads. For that reason, DeeZ, a newer alternative [67], implicitly assembles the underlying donor genome in order to encode these variants only once. A similar path is followed by CBC [133] and TSC [178]. All of these tools treat SAM fields in a separate manner, and apply a variety of compression techniques on each field.

Finally, Quip [85] and sam_comp [16] employ highly optimized statistical models for various SAM fields, which puts them among the best performing tools in terms of pure compression rate. The full description of the evaluated tools is available in the Chapter 2.

4.2 Criteria for Dataset and Tool Selection

4.2.1 General Criteria

Our goal in this study was to evaluate the majority of the available FASTQ and SAM compression tools in terms of (but not limited to) their compression performance, computational resources and correctness. In order to provide reproducible results and meaningful compression metrics, this section describes the benchmarking framework used by the authors both in terms of environment and genomic data used.

The following general criteria were used for the evaluation purposes:

Fairness and Reproducibility One important aspect of reviewing any set of software tools is the robustness and reproducibility of results. We wish this to be more than a review of current tools, but to also act as a framework for reviewing subsequent tools. In order to be as fair as possible, we have fully automated the benchmarking process by developing an open-source, freely available framework specifically tailored for NGS compression evaluation. All crashes were timely reported to the authors. Also, all output files were compared with the original files to check for mismatches or bugs. All crashes and mismatches are documented in Appendix B.

Data Diversity The data selected covers a variety of cases, both deep and shallow coverage, fixed length short reads (Illumina-style) and variable length long reads (PacBio-style), small and large genomes, and multiple sequencing experiment types (WGS, RNA-seq).

Format Compatibility We aim to investigate only those tools whose primary aim is to compress either FASTQ or SAM files. In this way, the performance and accuracy of each tool can be evaluated unambiguously.

Basic Capabilities Every tool has to provide a compression and fully functional decompression step in a reasonable time. Additional capabilities (e.g. random access, reference-free encoding, BAM slicing) were acknowledged but not evaluated in the detail.

Losslessness We limit ourselves to only lossless compression, accepting that controlled loss of data is appropriate in many cases but is outside the scope of this thesis. The slight exception to this rule is that we evaluate the tools which store the primary information losslessly (sequence, identifiers, qualities), but lose or change some of the derived data (e.g. FASTQ read order, comments, or SAM optional fields). All such cases are explicitly mentioned where appropriate.

4.2.2 Tool Selection Criteria

We aim to evaluate all available approaches used for HTS data compression. These include both industry-scale tools as well as research-oriented prototypes. A summary of the tools tested can be found in Table 4.1 and Appendix B.

While there are many other tools designed for storing the reads in both aligned and unaligned fashion, their main purpose is not compression *per se*, but a complete replacement of FASTQ or SAM schema with a special support for a particular use-cases (e.g. Hadoop compatibility etc.). For reference purposes, we have evaluated two tools from the latter category, ADAM [119] and Goby [20], and presented their perspectives in the Table 4.2.

Table 4.1: A summary of evaluated tools and their capabilities.

SAM	Random access	Reference required	Unsorted SAM	Statistics	Comments
Samtools	✓	No	✓	✓	
Sambamba	✓	No	✓	✓	
Picard	✓	No	✓	✓	
Cramtools	✓	Yes		✓	
Scramble	✓	Optional	✓	✓	Non-reference based encoding recommended for unsorted SAM
DeeZ	✓	Optional		✓	
CBC	✗	Yes			
TSC	✓	No			
Quip	✗	Optional			Reorders optional fields
sam_comp	✗	Optional			Does not support paired-end and optional fields
FASTQ	Random access	Reference required	Reordering		Comments
DSRC2	✗	No			
Fastqz	✗	Optional			No comments
Fqzcomp	✗	No			No comments
Slimfastq	✗	No			
FQC	✗	No			
LFQC	✗	No			
SCALCE	✗	No	✓		No comments
Orcom	✗	No	✓		Reads only
Mince	✗	No	✓		Reads only
BEETL	✓	No	✓		Reads only
LW-FQZip	✗	Yes			
Quip	✗	No			No comments
Leon	✗	No			No comments
k-Path	✗	Yes			Reads only
KIC	✗	No			

Compared to the other compression tools, their compression performance is significantly lower. Thus, we decided not to evaluate these tools in this benchmark.

Table 4.2: Performance of ADAM and Goby on MiSeq *E.coli* dataset. ADAM does not directly support SAM decompression.
The leftmost column indicates the overall compressed size. Middle column indicates the compression and decompression times relative to Samtools, with lower being better. The last column indicates the memory usage in MBs.

Sample			**DH10B**
Coverage			**490×**
Original	**5,733**		
Samtools	**1,440**	1.00	6.10
		1.00	6.20
ADAM	**1,218**	2.83	4,450.1
		N/A	N/A
Goby	**1,447**	2.18	3,031.30
		3.74	2,785.60

4.2.3 Dataset Selection Criteria

Investigations on the possibility to start an activity of formal standardization of genomic data representation is currently ongoing within ISO/IEC JTC1/SC29/Work Group 11, the Moving Picture Experts Group (MPEG) [82, 84]. One of the working items of this activity is the definition of a scientifically meaningful collection of genomic data samples including (i) both raw (FASTQ) and aligned (SAM) data, (ii) data from the most utilized sequencing technologies, (iii) data at different read depths (coverage), (iv) various types of experiments, and (v) various types of organisms.

Using a wide test bed covering various data characteristics can help identify specialized tools that perform very well on specific data types or fields in comparison to the more generic compression tools that perform uniformly on most samples or fields. It can also help us evaluate the robustness of algorithms that rely on statistical models for data sources. If a model utilized by a compression tool is too specific to a given data type, it may provide poor compression ratios on samples with different statistical properties (e.g. due to a different sequencing technology used to produce the data). Additionally, some tools are able to support both FASTQ and BAM file formats and (for some application scenarios) may thus be preferable to others with better compression ratios, but supporting only one format.

The selection of the data corpus was chosen to fit the following criteria without excessive duplication.

File format We have both FASTQ and SAM data sets. We could have used the same samples for both, requiring either that the FASTQ is aligned to produce the SAM file or the FASTQ file extracted from the SAM. Both of these have problems: producing

a SAM from FASTQ is very costly in CPU time and it is problematic to produce the exact same data every time due to regularly fluctuating software releases and the inherent stochastic nature of some alignment algorithms. Producing FASTQ from SAM is possible, but requires the data to be sorted back into the original (unknown) order as it came off the sequencing instruments. Given the wealth of public data, we instead chose to simply select datasets natively available in either FASTQ or SAM file format.

Genome size A small genome size lends itself well to observation of repeated fragments when using a small memory size (for example in an LZ style algorithm or *de novo* assembly). Larger genomes typically make these approaches less useful or require large amounts of CPU and memory. Therefore, our data set accounts for both small and large genomes.

Depth Sequence depth is defined as the average number of times each genomic location is covered by an aligned sequence fragment. Deep data offers good compression through redundancy. On data sorted by genomic location (typical for SAM) or with a small genome size, deep data offers easily observed DNA sequence redundancy. Shallow data will make redundancy less significant, in turn giving an advantage to reference based compression strategies.

Library type Sequencing libraries typically produce DNA fragments either singly or in pairs. Paired reads have common sequence names and typically the DNA fragments belonging to common pair align within a short distance of each other. Testing both types of data is useful to observe whether tools can take advantage of the partial redundancy in paired-end sequencing.

Instrument types The distribution of quality values, sequencing error rates, auxiliary tags present in SAM, lengths of DNA fragments and even whether the lengths are fixed or variable all differ between sequencing instruments. While the sequencing data sets are currently dominated by Illumina instruments, this may not always be the case and we need to observe whether software has been over-tuned for compressing only one type of data. Even within the Illumina data sets there is considerable variation as older instruments use approximately 40 distinct quality values while the newer HiSeq series quantize these to 8 values.

Experimental factors Other differences we wanted to capture included whole genome sequencing versus exome or RNA-Seq experiments with highly variable sequence depth, data from unknown or mixed genomes, and cancer samples where the rate of mutation is far higher than expected, both between sequences and the reference and potentially between different sequences themselves. Latter cases will have an impact on reference based compression and the robustness of deduplication techniques.

Used elsewhere Where possible, use data sets that have been previously used in papers.

4.2.4 Data set files

The initial dataset compiled by MPEG is composed of approximately 2.4 TB of human, human microbiome (metagenomic), bacterial and plant genome and transcriptome sequence data. In addition to normal human genome sequence data, genomic data from cancer cell lines is also included; see below for more details.

Note that since we started this study, the MPEG dataset has grown to approximately 4.5 TB, now including additional data types representing alternative sequencing technologies such as Oxford Nanopore, other species such as mouse or chimpanzee, and human genome sequences representing family members. Here we report our results on the initial MPEG dataset, since it was infeasible to rerun all the tools we tested (many of which required long running times for compression and decompression) every time a new data sample was introduced.

More samples are expected to be added to the MPEG dataset in near future. Subsequently, additional experiments will be carried out within the MPEG working group during the formal standardization activity, once the data set is complete and all candidate compression technologies are submitted for consideration [83].

Given the large size of the MPEG dataset we selected a subset for our benchmarks as follows:

SRR554369 FASTQ; paired short reads; Illumina GAIIx; 50× total depth; bacteria *P.aeruginosa*. Chosen as a small genome (6–7 MB) with medium depth. This dataset was used as a test set for LW-FQZip compression tool [189]. Available at ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR554/SRR554369/.

SRR327342 FASTQ; paired short reads; Illumina GAII; 175× total depth; yeast *S.cerevisiae*. Chosen as a small genome (12 MB) with high depth. This dataset was also used as a test set for LW-FQZip compression tool [189]. Available at ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR327/SRR327342/.

MH0001.081026 / ERA000116 FASTQ; paired short reads; Illumina GA; unknown depth; Human gut metagenome. Chosen as a dataset with mixed species and unknown references. Available at ftp://ftp.era.ebi.ac.uk/vol1/ERA000/ERA000116/fastq/MH0001_081026_clean.1.fq.gz and ftp://ftp.era.ebi.ac.uk/vol1/ERA000/ERA000116/fastq/MH0001_081026_clean.2.fq.gz

SRR1284073 FASTQ; single variable length long reads; PacBio; 140× depth; bacteria *E.Coli*. An example of a small genome (4.7 MB) covered by variable length long reads, but with low quality and a higher error rate. This is used as a test set for the

Jabba error correction tool [122]. Available at ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR128/003/SRR1284073/.

SRR870667 FASTQ; paired short reads; Illumina GAIIx; 35× total depth; plant *T.cacao*. This is the only plant genome in the corpus, with a medium sized genome (estimated 345 MB).

ERR174310 FASTQ; paired short reads; Illumina HiSeq 2000; 13× total depth; *H.sapiens* (NA12877) individual. This represents a common instrument, depth and organism. Available at ftp://ftp.sra.ebi.ac.uk/vol1/fastq/ERR174/ERR174310/.

ERP001775 FASTQ; paired short reads; Illumina; 120× depth; entire *H.sapiens* individual (NA12878) from the Illumina Platinum Genomes set. Due to the size, only fast tools are tested with this data.

DH10B SAM; paired short reads; Illumina MiSeq; 420× depth; bacteria *E.Coli*. Chosen for deep Illumina coverage of a small genome. Used in the DeeZ paper [67]. Available at ftp://webdata:webdata@ussd-ftp.illumina.com/Data/SequencingRuns/DH10B/MiSeq_Ecoli_DH10B_110721_PF.bam.

9827_2#49 / ERR317482 SAM; paired short reads; Illumina HiSeq 2000; 2× depth; *H.sapiens*. Extreme low uniform coverage of a large genome. Used in the Scramble [15] and Paridaens [137] papers. Available at ftp://ftp.sra.ebi.ac.uk/vol1/ERA242/ERA242167/bam/9827_2%2349.bam.

sample-2-12 / ERR303541 SAM; single variable length long reads; IonTorrent; 0.6× depth; *H.sapiens*. A single-ended run with variable mid-length reads. This is amplicon sequencing so has large variations in sequence depth across the genome. Available at ftp://ftp.sra.ebi.ac.uk/vol1/ERA229/ERA229587/bam/sample-2-12_sorted.bam.

K562.LID8465 SAM; paired short reads; Illumina; 6× depth; *H.sapiens*. An RNA-Seq experiment with variable depth Illumina data. Available at http://www.ebi.ac.uk/arrayexpress/files/E-MTAB-1728/K562_cytosol_LID8465_TopHat_v2.bam.

dm3 SAM; single variable length long reads; PacBio; 75×; bacteria *D.melanogaster*. Represents PacBio reads on a medium sized genome (168 MB). These are uncorrected reads, so they have low base accuracy and correspondingly complex and lengthy CIGAR strings. Available at http://bergmanlab.ls.manchester.ac.uk/data/tracks/dm3/dm3PacBio.bam.

HCC1954.mix1.n80t20 SAM; paired short reads; Illumina-like simulated data set; 30× depth; *H.sapiens*. This is an artificially mixed cancer cell line, used as part of the

TCGA cancer variant calling benchmark. This has a much higher degree of sequence variants than usual and was chosen as a good stress test for reference based compression tools. Available at https://cghub.ucsc.edu/datasets/benchmark_download.html (GeneTorrent is needed to download this sample).

NA12878.PB SAM; single variable length long reads; PacBio; 15× depth; *H.sapiens*. The widely sequenced NA12878 sample using variable length PacBio reads. Having the same sample sequenced with two very different instruments offers a good test bed for related work too, including evaluation of lossy compression. Available at ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/working/20131209_na12878_pacbio/si/NA12878.pacbio.bwa-sw.20140202.bam.

NA12878.S1 / ERR194146 SAM; paired short reads; Illumina HiSeq 2000; 50× depth; *H.sapiens*. See NA12878.PB for justification. Available at ftp://ftp.sra.ebi.ac.uk/vol1/ERA172/ERA172924/bam/NA12877_S1.bam. Due to the size, only fast tools are tested with this data.

4.3 Results

4.3.1 Experimental Setup

We have evaluated all of the tools on a dedicated machine with 6 AMD Opteron 8439 SE processors (each with 4 cores and 2.8 GHz clock rate) and 256 GB of memory. The machine was used only for benchmarking purposes during this study. In order to account for network interference, we performed all of the experiments on local storage. Caching side effects were minimized by clearing the cache before each invocation.

In order to evaluate the scalability of compression tools, we have evaluated all of the tools supporting multi-threading on configurations of 1, 4 and 8 working threads. Single-threaded mode is used as a reference, because many tools still do not exhibit parallelism efficiently. Because some tools do not allow single-threaded mode to be used (parallelism is hard-coded), we have forced every tool in single-threaded mode to use only one core via `cpubind` utility.

In addition to scalability, we have measured compression rations, memory usage, CPU load and decompression performance. Performance of separate fields was also measured, as long as it was available. Every tool was also tested for accuracy: does it decompress FASTQ or SAM file same as original, and if not, what are the differences.

4.3.2 FASTQ

An overview of our results is presented in the Table 4.4. Multi-threaded performance and memory usage are presented in Table 4.6. Graphical overview of the results is given in

Figure 4.1. The performance on paired-end libraries is presented in Table 4.9. Per-field statistics are presented in Table 4.5.

Some of the available tools achieve significant gains over Gzip and bzip2, both in terms of compression rate and speed. The best compression rates are offered by tools that reorder reads, which are especially effective for sequence compression. Alternatives such as LFQC may also provide good compression rates, but come with a high running time overhead. It should be noted that reordering based tools also perform very well in terms of speed. Their memory usage is slightly higher but not unreasonable and can be user-configured in most cases.

Table 4.6 shows the performance of tools when used in a multi-threaded environment. Many tools significantly improve their performance through parallelization, even though the performance improvement is not always proportional to the number of processors. The best trade-off between the number of processors and running time is typically achieved with four threads. Memory usage (with a few exceptions) is reasonable for most of the tools evaluated. Similar conclusions hold for SAM compression tools as well.

An additional observation is that the majority of the available tools are optimized for Illumina-style short, fixed-length reads—as can be seen, many tools do not provide an option to compress long, variable length read collections, such as PacBio data.

Special cases

For ERP001775 sample (with almost 1 TB of data), we have evaluated only those tools whose running time was reasonable, and which provided the reasonable compression rates (this decision was based on their performance on other samples). These tools were run with four threads, if it was possible—otherwise, single thread was used.

k-Path does not work properly if single-threaded mode is specified. Quip uses hard-coded multi-threading (with four threads); single-threaded mode was obtained by using `cpubind`. Fqzcomp, Fastqz, Slimfastq, FQC, LFQC, LW-FQZip and BEETL do not currently support parallelism. Mince was not tested in single-threaded mode on SRR870667 and ERR174310 samples due to the long running time. LFQC was not tested on ERR174310 due to the long running time.

Discussion

As can be seen from Table 4.4, the best sequence compression is obtained by reordering tools, in particular Orcom, Mince and SCALCE. However, note that Mince and Orcom do not support full FASTQ compression. Best tools which support whole FASTQ compression are LFQC, SCALCE and Slimfastq. However, LFQC's slow compression and decompression speed makes it impractical on any large dataset (e.g. decompression speed is on average 300× slower than Gzip's decompression speed).

Current state-of-the-art in quality score compression is Fqzcomp and its successor Slimfastq, whose probabilistic model offers the best performance on quality score data. As for read identifiers, tools which use probabilistic models, such are LFQC (based on PAQ), Quip and Slimfastq, again provide the best compression rates.

Note that the majority of arithmetic coding tools, such are Fqzcomp, Slimfastq, Quip and LFQC have rather high decompression times. Moreover, they usually do not exhibit parallelism in their current iterations. On the other side, reordering tools provide reasonable compression and decompression times. All evaluated tools (with the exception of Orcom, Mince and k-Path on large data sets) use reasonable amount of memory.

Taking everything into the consideration, reordering tools such are SCALCE or Orcom can be recommended as the best tools for FASTQ compression. If user prefers to keep the original ordering of FASTQ file, Slimfastq clearly presents the most adequate choice for FASTQ archival.

4.3.3 SAM

Detailed results are presented in the Table 4.4. Multi-threaded performance and memory usage are presented in Table 4.8. Graphical overview of the results is given in Figure 4.1. Per-field statistics are presented in Table 4.7.

Please note that we use Samtools as a reference point for BAM file format and Scramble as a reference point for CRAM file format. Other tools' implementation of these formats are mentioned separately if needed.

It is possible to obtain better compression rates than that achieved by Samtools, even with the simple use of Gzip. However, unlike Samtools, Gzip does not provide random access capability. Among the available tools, only CRAM family, DeeZ and TSC provide a random access facility. Interestingly, Scramble and DeeZ are able to improve sam_comp and Quip in most of the cases, both in terms of compression rate and speed, while additionally providing random access capability. Scramble is also able to decompress files faster than Samtools in most of the cases.

Special cases

For NA12878.S1 sample (around 600 GB of data), we have evaluated only those tools whose running time was reasonable, and which provided the reasonable compression rates (this decision was based on their performance on other samples). These tools were run with four threads, if it was possible.

Scramble, Cramtools, as well as Samtools can compress to CRAM file format. We used Cramtools to test CRAM v2 specification, while Scramble was used for CRAM v3 specification of the format.

KIC does not work properly if single-threaded mode is specified. Quip uses hard-coded multi-threading (with four threads); single-threaded mode was obtained by using `cpubind`. Samtools, Picard, Cramtools, TSC and sam_comp do not currently exhibit parallelism.

We have evaluated two versions of TSC—v1.2 for samples 9827.2.49, K652.LID8465 and HCC1954, and v1.4/v1.5 for samples DH10B, dm3 and sample-2-1 (v1.5 contains minor bugfixes over v1.4). This is due to the fact that v1.5 was not able to timely compress human samples. On the other hand, v1.2 does not support variable length reads.

Discussion

Based on results presented in the Table 4.4, overall winners are DeeZ and Scramble. DeeZ clearly offers the best compression rates (especially in bzip2 mode), while Scramble offers the best compression and decompression times. Both tools support multi-threading, and with 4 threads they are faster than Samtools both in compression and decompression. Furthermore, both tools support random access, and there is almost no reason to choose BAM over either DeeZ or Scramble. Similar to FASTQ tools, all evaluated tools use reasonable amount of memory.

As for quality scores, sam_comp provides the best compression rates since its model is based on Fqzcomp's quality score model. However, other models (e.g. those used by Scramble and DeeZ), which are conceptually simpler and faster in practice, offer reasonable compression rates in most of the cases, which makes them "good enough" for everyday use.

Random access

We also evaluated the random access performance on NA12878.S1 dataset for the tools which support random access. Three regions were picked as a random access performance indicator: (i) region within chr14 with no records, (ii) region within chrY with less than 2000 records, and (iii) whole chr22 region. Details are shown in the Table 4.3. For each region, we evaluated the following random access traits: (i) time necessary to decompress the requested region, (ii) number of seeks within compressed file, and (iii) number of bytes read from compressed file. Last two traits were obtained with the help of io_trace tool (`https://github.com/jkbonfield/io_trace`). We also included the index size for each file format.

As can be seen from the Table 4.3, all tools provide reasonable random access capabilities. It is interesting to note that Samtools (BAM), Scramble (CRAM) and DeeZ have different block sizes, ranging from few hundreds of records per block in BAM file format, to one million of records per block in the case of DeeZ. Larger block sizes require slightly longer decompression time in order to fetch smaller regions, but provide a smaller index size (and vice versa). Decompression of larger regions generally follows the observed decompression performance on whole files.

Table 4.3: A summary of random access performance for SAM/BAM tools which support it. If possible, all tools were run with 4 threads.

Tool	Index Size	Time (s)	Seeks	Bytes read (KB)
				chr14:107349000-107349540
Scramble	2.61	64	32,290	129,164
Samtools	8.35	0	3	25
Sambamba	8.35	0	6	123
DeeZ	0.65	1	9	687
				chrY:10000-20000
Scramble	2.61	1	2	520
Samtools	8.35	0	3	180
Sambamba	8.35	0	119	419
DeeZ	0.65	5	10	45,275
		Time (m:s)	Seeks	Bytes read (MB)
				chr22
Scramble	2.61	0:27	2	811
Samtools	8.35	1:53	3	1,367
Sambamba	8.35	0:25	555,240	1,367
DeeZ	0.65	1:55	27	705

Table 4.4: A summary of evaluated tools and their performance in a single-thread mode. (a) For each FASTQ tool and sample, the left two columns indicate the overall compressed size, and the sequence-only compressed size, respectively. Right column indicates the compression and decompression times relative to Gzip (pigz), with lower being better. The last three rows show the performance of sequence-only tools. In the ERR174324 sample, all tools were run with four threads, with the exception of pigz and Slimfastq. (b) For SAM tools, we only report here the overall size of the compressed data. Last row denotes sam_comp, which does not support all SAM fields. In the NA12878.S1 sample, all tools were run with four threads, with the exception of Samtools and sam_comp. sam_comp does not support all SAM fields, thus its performance was not compared to other tools.
Legend: Dark green colour indicates the best tool in the given category (compression ratio or time), while light green indicates second best tool. Analogously, orange denotes second worst tool, while magenta indicates the worst performance. Missing data points (either due to the crashes or compatibility issues) are marked with N/A.

Sample	SRR554369		SRR327342		MH0001.081026		SRR1284073		SRR870667		ERR174310		ERP001775	
Organism	*P.aeruginosa*		*S.cerevisiae*		*H.sapiens Gut*		*E.coli*		*T.cacao*		*H.sapiens*		*H.sapiens*	
Technology	Illumina GAIIx		Illumina GAII		Illumina GA		PacBio		Illumina GAIIx		HiSeq		HiSeq	
Coverage	25×		80×		Unknown		140×		20×		7×		120×	
Original	**550**		**3,881**		**1,880**		**1,309**		**22,944**		**53,869**		**2,717,029**	
	165		*947*		*512*		*649*		*7,463*		*20,966*		*1,059,387*	
pigz	**158**	1.00	**1,020**	1.00	**501**	1.00	**547**	1.00	**6,943**	1.00	**18,597**	1.00	**305,690**	1.00
	48	1.00	*277*	1.00	*149*	1.00	*188*	1.00	*2,108*	1.00	*5,982*	1.00	*104,927*	1.00
pbzip2	**125**	1.19	**831**	1.45	**390**	1.29	**463**	0.74	**5,577**	0.99	**14,887**	0.81	**242,834**	0.21
	44	5.97	*251*	6.85	*139*	6.35	*176*	6.99	*1,879*	3.61	*5,473*	2.83	*95,969*	1.23
DSRC2	**105**	0.22	**668**	0.26	**312**	0.24	**N/A**		**4,761**	0.21	**13,214**	0.20	**N/A**	
	41	2.11	*257*	3.09	*128*	1.91			*1,865*	1.39	*5,239*	1.22		
DSRC2 (extra)	**95**	0.71	**595**	0.70	**287**	0.80	**N/A**		**4,246**	0.70	**11,598**	0.67	**N/A**	
	39	11.81	*230*	11.33	*125*	12.96			*1,636*	6.25	*4,773*	5.78		
Fqzcomp	**89**	0.34	**559**	0.37	**280**	0.41	**N/A**		**4,028**	0.33	**11,320**	0.32	**N/A**	
	37	N/A	*203*	7.54	*120*	N/A			*1,556*	N/A	*4,623*	3.29		
Fqzcomp (extra)	**94**	0.39	**589**	0.39	**286**	0.41	**N/A**		**4,228**	0.35	**11,673**	0.34	**N/A**	
	41	N/A	*234*	7.50	*128*	N/A			*1,796*	N/A	*5,167*	3.47		
Fastqz	**N/A**		**N/A**		**N/A**		**N/A**		**N/A**		**10,955**	3.45	**N/A**	
											4,312	N/A		
Slimfastq	**94**	0.55	**507**	0.47	**266**	0.54	**N/A**		**4,280**	0.51	**11,045**	0.47	**178,092**	0.49
	30	11.46	*149*	9.55	*104*	11.32			*1,416*	5.80	*4,426*	4.76	*77,629*	5.94
FQC	**76**	1.05	**494**	1.18	**268**	1.39	**413**	0.98	**3,912**	1.16	**11,409**	1.22	**N/A**	
	N/A	11.78	N/A	12.87	N/A	17.07	N/A	12.12	N/A	5.93	N/A	5.74		
LFQC	**69**	18.63	**490**	18.54	**266**	21.06	**407**	18.03	**2,412**	14.46	**N/A**		**N/A**	
	17	315.41	*129*	310.81	*103*	339.93	*156*	386.25	N/A	N/A				

(a) FASTQ tools (part 1)

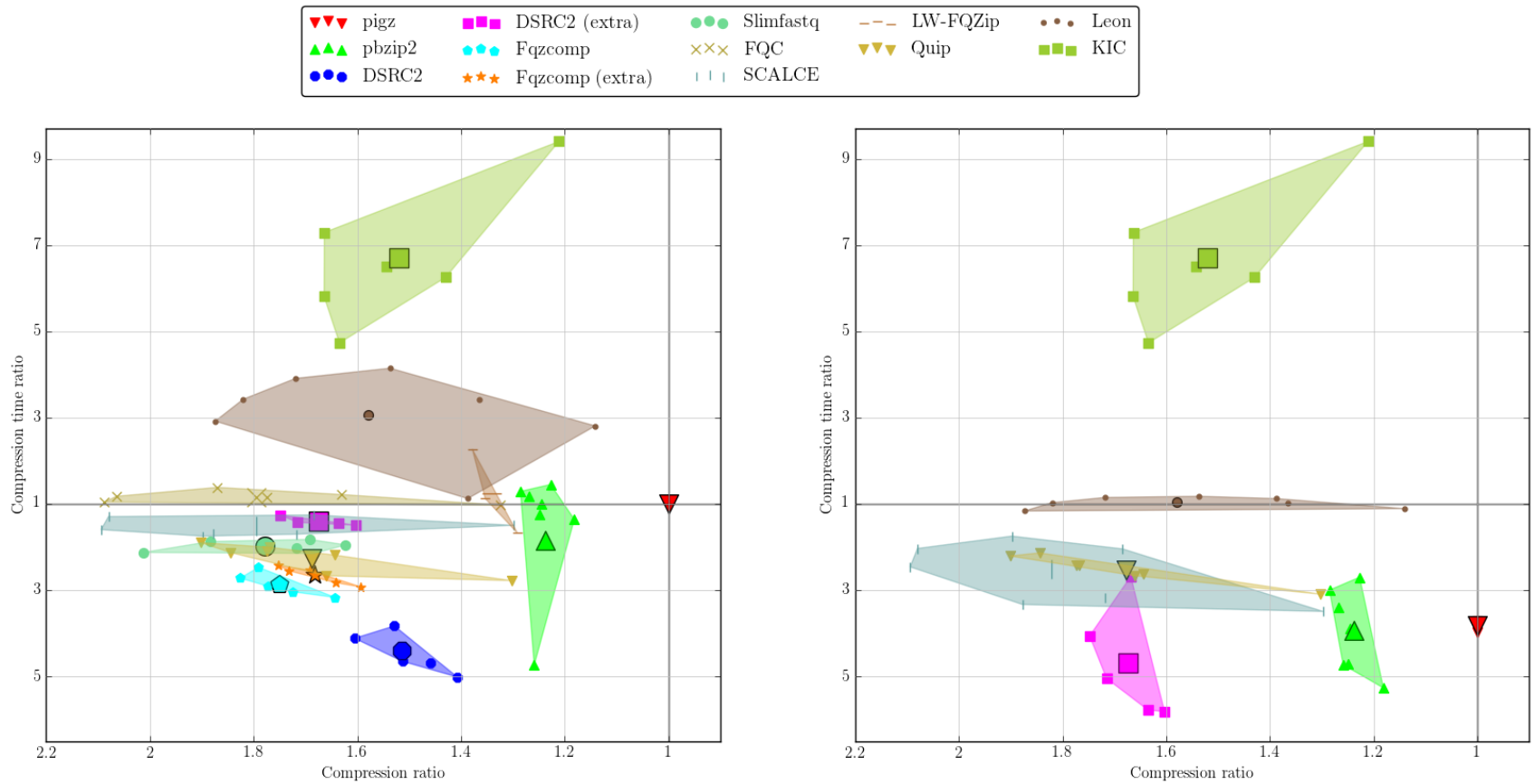
Sample	SRR554369		SRR327342		MH0001.081026		SRR1284073		SRR870667		ERR174310		ERP001775	
Organism	*P.aeruginosa*		*S.cerevisiae*		*H.sapiens Gut*		*E.coli*		*T.cacao*		*H.sapiens*		*H.sapiens*	
Technology	Illumina GAIIx		Illumina GAII		Illumina GA		PacBio		Illumina GAIIx		HiSeq		HiSeq	
Coverage	25×		80×		Unknown		140×		20×		7×		120×	
Original	**550**		**3,881**		**1,880**		**1,309**		**22,944**		**53,869**		**2,717,029**	
	165		*947*		*512*		*649*		*7,463*		*20,966*		*1,059,387*	
SCALCE	**76**	0.77	**487**	0.63	**297**	0.80	**421**	0.67	**3,699**	0.60	**10,827**	0.59	**161,067**	0.57
	17	9.05	*68*	8.23	*71*	12.17	*161*	9.78	*998*	4.89	*3,017*	4.57	*28,452*	1.94
LW-FQZip	**117**	1.13	**790**	0.60	**N/A**		**N/A**		**5,038**	2.27	**N/A**		**N/A**	
	45	5.62	*320*	5.16					*1,735*	2.50				
Quip	**89**	0.50	**537**	0.53	**272**	0.47	**420**	0.36	**3,914**	0.48	**11,312**	0.46	**184,051**	0.38
	37	10.70	*181*	11.53	*114*	11.37	*159*	10.59	*1,462*	5.57	*4,556*	5.22	*79.771*	4.64
Leon	**87**	3.43	**544**	2.92	**291**	3.91	**479**	2.81	**4,518**	4.15	**13,623**	3.43	**220,397**	1.13
	19	16.84	*89*	16.70	*87*	14.84	N/A	34.31	*1,360*	10.91	*4,739*	9.67	*83,539*	4.66
KIC	**95**	5.81	**613**	7.29	**307**	4.73	**451**	9.40	**4,498**	6.50	**13,006**	6.25	**N/A**	
	32	6.65	*188*	7.72	*122*	6.35	N/A	9.37	*1,594*	3.44	*4,915*	3.33		
Orcom		0.50		0.46		0.87	**N/A**			0.83		0.66		0.12
	11	1.51	*36*	0.91	*51*	1.87			*825*	1.22	*1,798*	0.83	*6,921*	0.23
BEETL		4.12		2.82		2.46	**N/A**			4.44		4.38	**N/A**	
	23	36.81	*117*	30.24	*114*	31.02			*1,200*	22.11	*3,912*	20.95		
Mince		4.52		4.26		5.19	**N/A**			2.42		2.57	**N/A**	
	10	2.38	*37*	2.24	*50*	3.25			*685*	0.86	*1,955*	0.90		
k-Path		2.03		1.73		13.04	**N/A**			2.29		3.22	**N/A**	
	14	30.08	*45*	20.19	*62*	149.57			*660*	15.39	*2,088*	16.57		

(a) FASTQ tools (part 2)

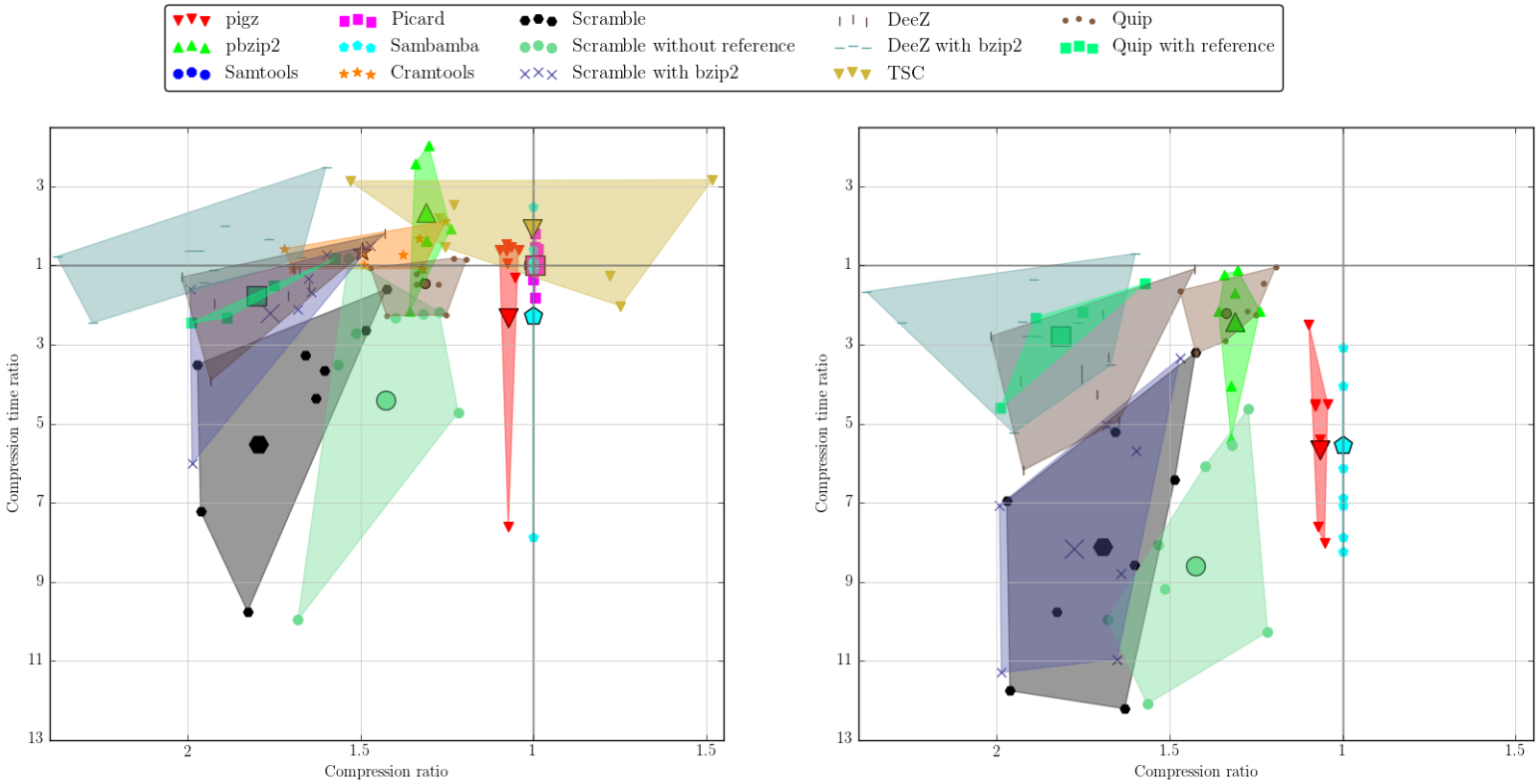
Sample	DH10B		9827.2.49		sample-2-1		K562.LID8465		dm3		NA12878.PB		HCC1954		NA12878.S1	
Organism	*E.coli*		*H.sapiens*		*H.sapiens*		*H.sapiens*		*D.melangoster*		*H.sapiens*		*H.sapiens*		*H.sapiens*	
Technology	MiSeq		HiSeq		IonTorrent		RNASeq		PacBio		PacBio		Cancer Cell		HiSeq	
Coverage	420×		2×		0.6×		6×		75×		15×		30×		50×	
Original	**5,579**		**21,059**		**5,924**		**75,915**		**30,081**		**126,545**		**427,028**		**589,083**	
pigz	**1,336**	0.77	**6,021**	1.55	**1,378**	1.48	**12,785**	1.06	**12,315**	1.39	**52,914**	1.37	**119,839**	1.40	**113,462**	0.13
		0.63		0.82		0.49		0.70		0.79		0.70		0.91		0.60
pbzip2	**1,074**	1.65	**5,243**	1.93	**1,127**	4.04	**10,251**	3.57	**9,717**	0.72	**43,128**	0.94	**100,280**	1.62	**89,598**	0.46
		3.16		3.39		3.72		2.46		2.93		3.94		3.23		0.59
Samtools	**1,407**	1.00	**6,499**	1.00	**1,469**	1.00	**13,757**	1.00	**12,853**	1.00	**57,090**	1.00	**131,566**	1.00	**121,710**	1.00
		1.00		1.00		1.00		1.00		1.00		1.00		1.00		1.00
Picard	**1,425**	1.42	**6,517**	1.04	**1,474**	1.82	**13,818**	1.48	**12,837**	0.74	**57,316**	0.55	**132,861**	1.18	**N/A**	
		2.76		1.52		2.10		2.44		1.09		1.00		1.91		
Sambamba	**1,407**	1.05	**6,499**	0.93	**1,469**	1.12	**13,757**	1.05	**12,859**	2.48	**57,090**	0.93	**131,566**	1.39	**121,710**	0.13
		1.08		1.13		0.97		0.97		1.92		1.12		1.12		0.53
Cramtools	**1,066**	0.93	**3,778**	1.42	**1,170**	2.12	**10,344**	1.70	**7,577**	0.93	**38,266**	1.01	**95,442**	1.28	**N/A**	
		1.71		1.67		4.93		2.00		2.05		2.39		1.50		
Scramble	**863**	0.23	**3,297**	0.29	**1,030**	0.62	**9,261**	0.38	**6,551**	0.14	**34,425**	0.31	**82,041**	0.27	**66,632**	0.10
		0.76		0.66		1.58		0.67		0.58		0.84		0.71		0.50
Scramble without reference	**899**	0.29	**4,236**	1.18	**1,113**	0.45	**9,839**	0.43	**10,562**	0.21	**44,843**	0.46	**86,914**	0.37	**72,407**	0.10
		0.74		0.63		1.06		0.78		1.14		1.54		0.79		0.47
Scramble with bzip2	**851**	0.76	**3,262**	0.62	**998**	1.50	**8,611**	1.27	**6,469**	0.17	**33,921**	0.48	**80,094**	0.60	**N/A**	
		0.89		0.66		1.72		0.81		0.67		1.63		0.82		
DeeZ	**823**	0.56	**3,221**	0.78	**1,028**	1.81	**8,120**	0.92	**6,681**	0.51	**34,639**	0.64	**78,473**	0.91	**62,966**	0.26
		3.90		2.46		5.51		3.35		1.77		1.86		2.94		1.00
DeeZ with bzip2	**730**	0.91	**2,734**	1.23	**918**	3.49	**7,266**	2.01	**6,585**	0.71	**34,172**	1.22	**74,509**	1.66	**53,497**	0.41
		10.11		5.60		9.86		7.91		4.86		6.67		6.39		1.90
TSC	**1,105**	2.21	**7,939**	0.80	**1,193**	2.55	**20,864**	3.17	**8,397**	3.14	**45,452**	1.46	**164,627**	0.50	**N/A**	
		9.05		2.24		6.75		6.27		6.46		6.43		2.65		
Quip	**1,103**	0.67	**4,419**	0.94	**1,230**	1.15	**11,186**	1.19	**9,024**	0.44	**42,642**	0.67	**98,303**	0.83	**97,165**	0.44
		10.69		7.81		4.43		8.27		7.52		9.87		9.05		2.18
Quip with reference	**803**	0.67	**N/A**		**N/A**		**8,743**	1.17	**6,461**	0.41	**N/A**		**N/A**		**64,493**	0.43
		10.06						8.20		7.19						2.20
sam_comp	**700**	0.68	**2,649**	0.76	**891**	1.20	**7,023**	0.71	**8,356**	0.51	**32,670**	0.59	**42,522**	0.62	**53,263**	0.37
		3.36		2.95		6.54		3.56		5.49		5.42		3.25		2.00

(b) SAM tools

Figure 4.1: Visualised single-threaded and 4-threaded performance of FASTQ and SAM tools' compression rate and compression speed. Centre of the coordinate system represents the Gzip's (pigz) or Samtools' performance. On the x-axis, left side represents the compression gain over pigz (Samtools), while right side denotes the compression loss, measured in multiples of pigz's (Samtools') size. For example, 1.5 on the left side in the SAM plot indicates that the compression ratio is 1.5× higher than Samtools. On the y-axis, upper part indicates the slowdown, while lower part indicates speed-up. Each point represents the performance of one tool on one sample; shaded polygon represents the performance of one tool across the all samples. For each such polygon, its centroid (representing average performance of the tool) is shown as a large point in the middle.



(a) FASTQ tools


pigz
pbzip2
Samtools
Picard
Sambamba
Cramtools
Scramble
Scramble without reference
Scramble with bzip2
DeeZ
DeeZ with bzip2
TSC
Quip
Quip with reference
Compression time ratio
Compression ratio
3
1
3
5
7
9
11
13
2
1.5
1
1.5

(b) SAM tools

Table 4.5: Performance of FASTQ tools on various FASTQ fields. Left side shows size in MB, while right side shows the compression ratio. 1 MB is calculated as 10^9 bytes. Colour codes are described in Table 4.4.

Sample	SRR554369		SRR327342		MH0001.081026		SRR1284073		SRR870667		ERR174310		ERP001775	
Original	165	1.00	947	1.00	512	1.00	649	1.00	7,463	1.00	20,966	1.00	368,183	1.00
pigz	48	0.29	277	0.29	149	0.29	188	0.29	2,108	0.28	5,982	0.29	104,927	0.28
pbzip2	44	0.27	251	0.27	139	0.27	176	0.27	1,879	0.25	5,473	0.26	95,969	0.26
DSRC2	41	0.25	257	0.27	128	0.25	N/A		1,865	0.25	5,239	0.25	N/A	
DSRC2 (extra)	39	0.24	230	0.24	125	0.24	N/A		1,636	0.22	4,773	0.23	N/A	
Fqzcomp	37	0.22	203	0.21	120	0.23	N/A		1,556	0.21	4,623	0.22	N/A	
Fqzcomp (extra)	41	0.25	234	0.25	128	0.25	N/A		1,796	0.24	5,167	0.25	N/A	
Slimfastq	30	0.18	149	0.16	104	0.20	N/A		1,416	0.19	4,426	0.21	77,629	0.21
LFQC	17	0.10	129	0.14	103	0.20	156	0.24	N/A		N/A		N/A	
SCALCE	17	0.10	68	0.07	71	0.14	161	0.25	998	0.13	3,017	0.14	28,452	0.08
LW-FQZip	45	0.27	320	0.34	N/A		N/A		1,735	0.23	N/A		N/A	
Quip	37	0.22	181	0.19	114	0.22	159	0.25	1,462	0.20	4,556	0.22	79,771	0.22
Leon	19	0.11	89	0.09	87	0.17	N/A		1,360	0.18	4,739	0.23	83,539	0.23
KIC	32	0.19	188	0.20	122	0.24	N/A		1,594	0.21	4,915	0.23	N/A	
Orcom	11	0.06	36	0.04	51	0.10	N/A		825	0.11	1,798	0.09	6,921	0.02
BEETL	23	0.14	117	0.12	114	0.22	N/A		1,200	0.16	3,912	0.19	N/A	
Mince	9.98	0.06	37	0.04	50	0.10	N/A		685	0.09	1,955	0.09	N/A	
k-Path	14	0.08	45	0.05	62	0.12	N/A		660	0.09	2,088	0.10	N/A	

(a) Read sequences

Sample	SRR554369		SRR327342		MH0001.081026		SRR1284073		SRR870667		ERR174310		ERP001775	
Original	165	1.00	947	1.00	512	1.00	649	1.00	7,463	1.00	20,966	1.00	368,183	1.00
pigz	65	0.39	426	0.45	184	0.36	309	0.48	3,027	0.41	8,639	0.41	133,526	0.36
pbzip2	57	0.35	400	0.42	167	0.33	283	0.44	2,724	0.37	7,428	0.35	112,046	0.30
DSRC2	57	0.35	389	0.41	168	0.33	N/A		2,699	0.36	7,489	0.36	N/A	
DSRC2 (extra)	49	0.30	344	0.36	145	0.28	N/A		2,350	0.31	6,253	0.30	N/A	
Fqzcomp	48	0.29	334	0.35	143	0.28	N/A		2,292	0.31	6,288	0.30	N/A	
Fqzcomp (extra)	48	0.29	333	0.35	142	0.28	N/A		2,253	0.30	6,097	0.29	N/A	
Slimfastq	60	0.36	334	0.35	145	0.28	N/A		2,733	0.37	6,287	0.30	94,662	0.26
LFQC	48	0.29	341	0.36	147	0.29	250	0.39	2,290	0.31	N/A		N/A	
SCALCE	52	0.32	349	0.37	150	0.29	260	0.40	2,362	0.32	6,738	0.32	102,336	0.28
LW-FQZip	63	0.38	435	0.46	N/A		N/A		3,001	0.40	N/A		N/A	
Quip	48	0.29	334	0.35	142	0.28	259	0.40	2,249	0.30	6,284	0.30	95,949	0.26
Leon	64	0.39	429	0.45	184	0.36	N/A		3,018	0.40	8,533	0.41	130,578	0.35
KIC	56	0.34	384	0.41	163	0.32	N/A		2,626	0.35	7,394	0.35	N/A	

(b) Quality scores

Sample	SRR554369		SRR327342		MH0001.081026		SRR1284073		SRR870667		ERR174310		ERP001775	
Original	107	1.00	963	1.00	405	1.00	4.72	1.00	3,871	1.00	10,900	1.00	189,618	1.00
pigz	14	0.13	99	0.10	41	0.10	0.82	0.17	528	0.14	1,570	0.14	27,555	0.15
pbzip2	12	0.11	71	0.07	32	0.08	0.68	0.14	421	0.11	1,223	0.11	21,531	0.11
DSRC2	6.23	0.06	21	0.02	16	0.04	N/A		197	0.05	486	0.04	N/A	
DSRC2 (extra)	6.22	0.06	21	0.02	17	0.04	N/A		260	0.07	572	0.05	N/A	
Fqzcomp	4.79	0.04	21	0.02	16	0.04	N/A		179	0.05	408	0.04	N/A	
Fqzcomp (extra)	4.79	0.04	21	0.02	16	0.04	N/A		179	0.05	408	0.04	N/A	
Slimfastq	3.79	0.04	23	0.02	17	0.04	N/A		128	0.03	323	0.03	5,699	0.03
LFQC	3.62	0.03	21	0.02	16	0.04	0.26	0.05	122	0.03	N/A		N/A	
SCALCE	6.78	0.06	70	0.07	77	0.19	0.57	0.12	341	0.09	1,074	0.10	20,544	0.11
LW-FQZip	8.61	0.08	35	0.04	N/A		N/A		302	0.08	N/A		N/A	
Quip	4.89	0.05	21	0.02	16	0.04	0.28	0.06	203	0.05	472	0.04	8,326	0.04
Leon	4.06	0.04	26	0.03	20	0.05	N/A		139	0.04	355	0.03	6,263	0.03
KIC	7.14	0.07	41	0.04	22	0.05	N/A		278	0.07	697	0.06	N/A	

(c) Read identifiers

Table 4.6: Time and memory performance of FASTQ tools for 1, 4 and 8 threads. For each tool, first row shows its compression performance, while second row shows its decompression performance. Time is measured as a fraction of Gzip's performance (in this case, pigz in single-threaded mode). Memory performance is presented in MBs.

Sample	SRR554369			SRR327342			MH0001.081026			SRR1284073			SRR870667			ERR174310			ERP001775	
Threads	1	4	8	1	4	8	1	4	8	1	4	8	1	4	8	1	4	8	1	4
pigz	1.00	0.25	0.13	1.00	0.26	0.15	1.00	0.27	0.13	1.00	0.26	0.13	1.00	0.26	0.15	1.00	0.26	0.14	1.00	0.27
	1.00	0.92	0.81	1.00	0.92	0.98	1.00	0.89	0.99	1.00	1.00	0.93	1.00	0.78	0.85	1.00	0.78	0.70	1.00	0.96
pbzip2	1.19	0.30	0.16	1.45	0.37	0.21	1.29	0.33	0.17	0.74	0.19	0.10	0.99	0.26	0.15	0.81	0.21	0.12	N/A	0.21
	5.97	1.70	0.89	6.85	2.75	0.95	6.35	1.50	0.80	6.99	1.75	0.97	3.61	1.06	0.83	2.83	1.10	0.81	N/A	1.23
DSRC2	0.22	0.06	0.03	0.26	0.06	0.04	0.24	0.06	0.04	N/A	N/A	N/A	0.21	0.06	0.05	0.20	0.06	0.04	N/A	N/A
	2.11	0.59	0.35	3.09	0.63	0.34	1.91	1.03	1.03	N/A	N/A	N/A	1.39	0.79	0.78	1.22	0.70	0.54	N/A	N/A
DSRC2 (extra)	0.71	0.37	0.38	0.70	0.20	0.11	0.80	0.25	0.16	N/A	N/A	N/A	0.70	0.17	0.09	0.67	0.17	0.09	N/A	N/A
	11.81	6.00	6.05	11.33	3.06	1.67	12.96	5.07	3.99	N/A	N/A	N/A	6.25	1.60	0.84	5.78	1.49	0.75	N/A	N/A
Fqzcomp	0.34			0.37			0.41			N/A			0.33			0.32			N/A	
	N/A			7.54			N/A			N/A			N/A			3.29			N/A	
Fqzcomp (extra)	0.39			0.39			0.41			N/A			0.35			0.34			N/A	
	N/A			7.50			N/A			N/A			N/A			3.47			N/A	
Fastqz	N/A			N/A			N/A			N/A			N/A			3.45			N/A	
	N/A			N/A			N/A			N/A			N/A			N/A			N/A	
Slimfastq	0.55			0.47			0.54			N/A			0.51			0.47			0.49	
	11.46			9.55			11.32			N/A			5.80			4.76			5.94	
FQC	1.05			1.18			1.39			0.98			1.16			1.22			N/A	
	11.78			12.87			17.07			12.12			5.93			5.74			N/A	
LFQC	18.63			18.54			21.06			18.03			14.46			N/A			N/A	
	315.41			310.81			339.93			386.25			N/A			N/A			N/A	
SCALCE	0.77	0.49	0.46	0.63	0.41	0.38	0.80	0.49	0.46	0.67	0.29	0.19	0.60	0.30	0.27	0.59	0.32	0.29	N/A	0.57
	9.05	3.19	2.14	8.23	2.86	1.86	12.17	4.31	3.09	9.78	4.26	2.38	4.89	1.64	1.16	4.57	1.62	1.14	N/A	1.94
LW-FQZip	1.13			0.60			N/A			N/A			2.27			N/A			N/A	
	5.62			5.16			N/A			N/A			2.50			N/A			N/A	
Quip	0.50	0.41		0.53	0.45		0.47	0.47		0.36	0.32		0.48	0.41		0.46	0.38		N/A	0.38
	10.70	7.32		11.53	8.33		11.37	9.21		10.59	10.08		5.57	4.07		5.22	3.84		N/A	4.64
Leon	3.43	1.03	0.70	2.92	0.86	0.55	3.91	1.15	0.72	2.81	0.90	0.81	4.15	1.18	0.73	3.43	1.02	0.65	N/A	1.13
	16.84	6.51	4.27	16.70	6.92	4.24	14.84	5.67	3.88	34.31	11.45	11.39	10.91	4.33	2.64	9.67	4.00	2.53	N/A	4.66
KIC	N/A	5.81	6.17	N/A	7.29	7.71	N/A	4.73	4.92	N/A	9.40	9.34	N/A	6.50	6.19	N/A	6.25	6.53	N/A	N/A
	N/A	6.65	8.32	N/A	7.72	7.22	N/A	6.35	7.03	N/A	9.37	9.65	N/A	3.44	3.47	N/A	3.33	3.37	N/A	N/A
Orcom	0.50	0.19	0.15	0.46	0.12	0.08	0.87	0.23	0.14	N/A	N/A	N/A	0.83	0.21	0.13	0.66	0.16	0.09	N/A	0.12
	1.51	0.49	0.27	0.91	0.24	0.13	1.87	0.53	0.28	N/A	N/A	N/A	1.22	0.32	0.24	0.83	0.26	0.25	N/A	0.23
BEETL	4.12			2.82			2.46			N/A			4.44			4.38			N/A	
	36.81			30.24			31.02			N/A			22.11			20.95			N/A	
Mince	4.52	1.72	1.13	4.26	1.57	1.04	5.19	2.46	2.01	N/A	N/A	N/A	N/A	2.42	1.35	N/A	2.57	1.42	N/A	N/A
	2.38	2.00	2.03	2.24	1.98	1.93	3.25	2.91	3.46	N/A	N/A	N/A	N/A	0.86	0.87	N/A	0.90	0.91	N/A	N/A
k-Path	N/A	2.03	1.89	N/A	1.73	1.64	N/A	13.04	13.69	N/A	N/A	N/A	N/A	2.29	2.20	N/A	3.22	3.00	N/A	N/A
	N/A	30.08	29.92	N/A	20.19	20.55	N/A	149.57	150.72	N/A	N/A	N/A	N/A	15.39	15.14	N/A	16.57	16.50	N/A	N/A

(a) Time performance

Sample			SRR554369			SRR327342			MH0001.081026			SRR1284073			SRR870667			ERR174310			ERP001775	
Threads	1	4	8	1	4	8	1	4	8	1	4	8	1	4	8	1	4	8	1	4
pigz	5.80	5.90	7.80	5.80	5.90	7.90	5.70	5.70	8.00	5.70	5.70	10.10	5.80	5.90	10.60	5.70	7.50	8.70	5.90	9.50
	5.90	5.90	6.00	5.90	5.90	6.00	5.70	5.70	5.80	5.70	5.70	5.70	5.80	5.90	5.90	5.70	5.70	5.70	5.90	5.80
pbzip2	10.40	37.30	70.80	11.40	39.30	74.20	11.40	37.70	71.50	10.60	38.40	74.30	12.00	40.30	74.20	12.00	39.70	75.90	N/A	41.00
	5.90	24.60	46.40	5.90	25.30	46.70	5.70	25.10	46.30	5.70	27.40	49.90	5.80	25.70	48.10	5.70	27.40	48.50	N/A	27.00
DSRC2	17.00	170.70	153.60	17.60	165.10	139.50	18.60	170.50	170.30	N/A	N/A	N/A	17.40	165.20	158.00	20.00	186.40	358.80	N/A	N/A
	15.90	119.90	237.30	19.70	139.40	257.10	19.90	137.80	257.30	N/A	N/A	N/A	20.70	140.30	285.80	29.40	208.00	400.20	N/A	N/A
DSRC2 (extra)	417.40	751.70	754.10	451.40	2,721.60	4,892.30	434.70	2,325.90	2,669.50	N/A	N/A	N/A	437.40	2,622.90	5,235.90	485.60	2,706.30	5,346.00	N/A	N/A
	367.20	789.60	790.40	405.90	2,794.00	4,464.40	398.70	2,468.50	2,764.50	N/A	N/A	N/A	374.30	3,103.30	5,923.10	392.50	3,015.60	6,009.90	N/A	N/A
Fqzcomp	77.50			79.50			79.50			N/A			75.50			77.50			N/A	
	N/A			65.40			N/A			N/A			N/A			65.40			N/A	
Fqzcomp (extra)	326.50			328.50			328.60			N/A			326.50			330.50			N/A	
	N/A			314.40			N/A			N/A			N/A			318.40			N/A	
Fastqz	N/A			N/A			N/A			N/A			N/A			1,527.30			N/A	
	N/A			N/A			N/A			N/A			N/A			N/A			N/A	
Slimfastq	79.00			79.00			79.00			N/A			79.00			79.00			79.00	
	78.90			78.90			78.90			N/A			78.90			78.90			78.90	
FQC	428.10			682.90			682.20			681.40			683.80			682.70			N/A	
	66.30			120.30			322.20			67.60			274.80			163.00			N/A	
LFQC	2,120.10			3,762.10			3,648.40			3,258.90			3,762.50			N/A			N/A	
	2,014.70			3,328.20			3,327.80			2,824.60			N/A			N/A			N/A	
SCALCE	1,381.70	1,381.00	1,400.30	2,923.70	2,921.60	2,939.30	2,431.60	2,520.70	2,538.80	1,897.90	1,899.60	1,932.00	5,295.50	5,296.40	5,313.00	5,322.10	5,321.20	5,335.60	N/A	5,327.40
	1,008.90	1,047.60	1,118.00	1,007.00	1,054.70	1,117.70	1,006.00	1,054.70	1,103.90	1,009.30	1,057.50	1,121.40	1,007.60	1,054.50	1,104.40	1,006.90	1,055.40	1,104.20	N/A	1,055.90
LW-FQZip	197.60			191.90			N/A			N/A			687.20			N/A			N/A	
	22.00			103.10			N/A			N/A			674.20			N/A			N/A	
Quip	391.00	389.20		393.00	393.60		395.90	394.90		632.50	632.30		396.60	397.80		399.00	398.60		N/A	403.30
	387.30	387.30		391.40	391.40		391.90	393.90		636.70	636.70		388.40	387.50		389.60	390.40		N/A	394.80
Leon	72.80	208.50	392.20	104.90	268.70	446.20	90.10	165.20	300.80	1,642.20	2,747.60	3,101.30	629.70	1,131.20	1,762.30	3,253.60	3,400.40	3,658.40	N/A	3,651.90
	64.00	135.50	217.00	69.10	131.10	216.80	47.20	94.20	154.10	1,691.90	2,752.50	2,755.20	505.10	602.60	730.70	2,898.00	2,898.90	2,990.90	N/A	3,154.70
KIC	N/A	1,932.80	1,968.50	N/A	2,028.70	1,964.30	N/A	1,939.10	1,962.80	N/A	2,108.10	2,108.60	N/A	2,094.60	2,124.80	N/A	2,127.90	2,160.40	N/A	N/A
	N/A	841.50	836.60	N/A	864.10	866.60	N/A	866.80	874.00	N/A	922.20	916.30	N/A	877.40	912.90	N/A	912.30	917.70	N/A	N/A
Orcom	403.00	768.80	768.00	396.40	1,519.20	1,420.10	440.10	1,370.80	1,393.80	N/A	N/A	N/A	453.00	1,230.20	1,884.30	527.20	4,851.80	2,635.30	N/A	20,892.90
	11.40	67.10	124.30	25.10	131.40	214.70	24.00	77.30	124.30	N/A	N/A	N/A	701.30	1,892.70	2,954.00	547.00	2,715.90	4,530.80	N/A	42,584.10
BEETL	7.40			7.70			7.70			N/A			7.70			7.70			N/A	
	27.90			224.40			169.30			N/A			1,093.10			3,246.90			N/A	
Mince	1,142.20	1,168.60	1,200.50	4,800.80	4,825.80	4,956.20	5,684.10	5,710.80	5,845.70	N/A	N/A	N/A	N/A	23,654.70	24,139.00	N/A	68,147.40	69,341.40	N/A	N/A
	242.60	242.60	242.60	242.60	242.60	242.60	242.60	242.60	242.60	N/A	N/A	N/A	N/A	242.60	242.60	N/A	242.60	242.60	N/A	N/A
k-Path	N/A	2,264.10	2,264.30	N/A	6,381.00	6,846.30	N/A	46,909.50	37,332.00	N/A	N/A	N/A	N/A	33,195.40	36,339.50	N/A	85,477.10	84,725.00	N/A	N/A
	N/A	1,539.20	1,528.80	N/A	3,979.10	4,131.40	N/A	47,391.20	47,521.70	N/A	N/A	N/A	N/A	25,558.00	25,834.10	N/A	53,502.10	51,020.30	N/A	N/A

(b) Memory performance

Table 4.7: Performance of SAM tools on various SAM fields. Left side shows size in MB, while right side shows the compression ratio. Colour codes are described in Table 4.4. (a) Read sequences include SEQ, CIGAR, RNAME and POS fields. (b) Quality scores include only QUAL field. (c) Read identifiers consist of QNAME. (d) Auxiliary fields consist of all other SAM fields, including the optional fields. 1 MB is calculated as 10^9 bytes.

Sample	DH10B		9827.2.49		sample-2-1		K562.LID8465		dm3		NA12878.PB		HCC1954		NA12878.S1	
Original	2,290	1.00	6,429	1.00	2,203	1.00	22,935	1.00	17,497	1.00	57,660	1.00	107,915	1.00	185,637	1.00
pigz	57	0.03	1,098	0.17	65	0.03	484	0.02	4,848	0.28	15,272	0.26	8,626	0.08	9,848	0.05
pbzip2	80	0.03	1,351	0.21	57	0.03	635	0.03	3,826	0.22	13,889	0.24	11,814	0.11	14,945	0.08
Samtools	54	0.02	1,198	0.19	79	0.04	480	0.02	4,846	0.28	15,543	0.27	9,066	0.08	10,315	0.06
Cramtools (CRAM v2)	45	0.02	218	0.03	62	0.03	568	0.02	1,532	0.09	7,952	0.14	6,543	0.06	N/A	
Scramble (CRAM v3)	18	0.01	136	0.02	38	0.02	154	0.01	1,367	0.08	6,753	0.12	4,071	0.04	3,567	0.02
Scramble without reference	51	0.02	975	0.15	97	0.04	543	0.02	4,978	0.28	15,574	0.27	7,587	0.07	7,697	0.04
Scramble with bzip2	18	0.01	132	0.02	33	0.02	142	0.01	1,361	0.08	6,680	0.12	4,057	0.04	N/A	
DeeZ	20	0.01	144	0.02	34	0.02	187	0.01	1,575	0.09	7,303	0.13	4,112	0.04	3,566	0.02
DeeZ with bzip2	20	0.01	138	0.02	34	0.02	184	0.01	1,554	0.09	7,241	0.13	3,903	0.04	3,018	0.02
TSC	26	0.01	1,132	0.18	46	0.02	663	0.03	2,400	0.14	15,228	0.26	8,604	0.08	N/A	
Quip	367	0.16	1,538	0.24	226	0.10	3,620	0.16	3,932	0.22	13,087	0.23	25,387	0.24	41,813	0.23
Quip with reference	68	0.03	N/A		N/A		1,177	0.05	1,369	0.08	N/A		N/A		9,141	0.05
sam_comp	21	0.01	122	0.02	38	0.02	256	0.01	3,347	0.19	9,383	0.16	3,716	0.03	2,782	0.01

(a) Read sequences and their mapping information

Sample	DH10B		9827.2.49		sample-2-1		K562.LID8465		dm3		NA12878.PB		HCC1954		NA12878.S1	
Original	1,976	1.00	5,646	1.00	1,930	1.00	18,732	1.00	12,307	1.00	44,123	1.00	95,182	1.00	159,860	1.00
pigz	876	0.44	3,027	0.54	976	0.51	7,428	0.40	5,941	0.48	25,802	0.58	38,942	0.41	61,576	0.39
pbzip2	773	0.39	2,835	0.50	902	0.47	6,756	0.36	5,618	0.46	24,692	0.56	36,681	0.39	53,021	0.33
Samtools	876	0.44	3,027	0.54	976	0.51	7,428	0.40	5,941	0.48	25,800	0.58	38,969	0.41	61,575	0.39
Cramtools (CRAM v2)	892	0.45	3,065	0.54	986	0.51	7,588	0.41	6,007	0.49	25,805	0.58	39,750	0.42	N/A	
Scramble (CRAM v3)	714	0.36	2,673	0.47	868	0.45	6,444	0.34	5,140	0.42	23,202	0.53	33,376	0.35	50,498	0.32
Scramble without reference	714	0.36	2,674	0.47	871	0.45	6,446	0.34	5,562	0.45	24,769	0.56	33,382	0.35	50,519	0.32
Scramble with bzip2	714	0.36	2,673	0.47	868	0.45	6,444	0.34	5,067	0.41	23,202	0.53	33,376	0.35	N/A	
DeeZ	713	0.36	2,685	0.48	875	0.45	6,555	0.35	5,069	0.41	23,150	0.52	33,517	0.35	50,322	0.31
DeeZ with bzip2	631	0.32	2,241	0.40	794	0.41	5,750	0.31	4,996	0.41	22,938	0.52	31,011	0.33	42,255	0.26
TSC	884	0.45	2,728	0.48	990	0.51	6,664	0.36	5,944	0.48	25,836	0.59	34,283	0.36	N/A	
Quip	634	0.32	2,361	0.42	788	0.41	5,994	0.32	5,029	0.41	22,821	0.52	30,831	0.32	43,814	0.27
Quip with reference	634	0.32	N/A		N/A		5,994	0.32	5,029	0.41	N/A		N/A		43,814	0.27
sam_comp	619	0.31	2,230	0.39	779	0.40	5,685	0.30	4,990	0.41	22,895	0.52	30,779	0.32	41,851	0.26

(b) Quality scores

Sample	DH10B		9827.2.49		sample-2-1		K562.LID8465		dm3		NA12878.PB		HCC1954		NA12878.S1	
Original	234	1.00	1,779	1.00	297	1.00	9,169	1.00	110	1.00	1,983	1.00	32,073	1.00	63,358	1.00
pigz	64	0.27	306	0.17	91	0.31	1,734	0.19	15	0.13	231	0.12	5,720	0.18	9,460	0.15
pbzip2	56	0.24	270	0.15	66	0.22	1,281	0.14	11	0.10	179	0.09	4,884	0.15	8,050	0.13
Samtools	68	0.29	320	0.18	92	0.31	1,803	0.20	15	0.14	243	0.12	5,959	0.19	9,873	0.16
Cramtools (CRAM v2)	70	0.30	311	0.17	94	0.32	1,772	0.19	16	0.14	241	0.12	6,064	0.19	N/A	
Scramble (CRAM v3)	65	0.28	288	0.16	93	0.31	1,655	0.18	16	0.14	230	0.12	5,760	0.18	9,291	0.15
Scramble without reference	65	0.28	288	0.16	93	0.31	1,655	0.18	16	0.14	230	0.12	5,760	0.18	9,291	0.15
Scramble with bzip2	59	0.25	275	0.15	69	0.23	1,345	0.15	14	0.12	199	0.10	4,955	0.15	N/A	
DeeZ	44	0.19	203	0.11	77	0.26	925	0.10	13	0.12	200	0.10	4,853	0.15	6,516	0.10
DeeZ with bzip2	41	0.17	192	0.11	57	0.19	919	0.10	11	0.10	168	0.08	4,239	0.13	6,078	0.10
TSC	67	0.29	542	0.30	93	0.31	2,034	0.22	15	0.14	234	0.12	10,309	0.32	N/A	
Quip	55	0.24	262	0.15	67	0.23	1,019	0.11	27	0.25	499	0.25	5,745	0.18	7,768	0.12
Quip with reference	55	0.24	N/A		N/A		1,019	0.11	27	0.25	N/A		N/A		7,768	0.12
sam_comp	55	0.24	275	0.15	65	0.22	1,013	0.11	19	0.17	373	0.19	7,538	0.24	8,074	0.13

(c) Read identifiers

Sample	DH10B		9827.2.49		sample-2-1		K562.LID8465		dm3		NA12878.PB		HCC1954		NA12878.S1	
Original	921	1.00	6,527	1.00	1,284	1.00	22,121	1.00	149	1.00	22,466	1.00	180,549	1.00	161,234	1.00
pigz	118	0.13	608	0.09	61	0.05	1,078	0.05	37	0.25	4,126	0.18	49,809	0.28	10,107	0.06
pbzip2	84	0.09	463	0.07	38	0.03	806	0.04	27	0.18	3,827	0.17	41,071	0.23	8,535	0.05
Samtools	117	0.13	555	0.09	56	0.04	1,012	0.05	35	0.24	4,177	0.19	48,972	0.27	9,536	0.06
Cramtools (CRAM v2)	57	0.06	174	0.03	27	0.02	391	0.02	20	0.14	4,239	0.19	42,979	0.24	N/A	
Scramble (CRAM v3)	64	0.07	178	0.03	30	0.02	639	0.03	20	0.14	4,153	0.18	38,323	0.21	2,896	0.02
Scramble without reference	67	0.07	276	0.04	50	0.04	820	0.04	23	0.15	4,183	0.19	39,723	0.22	4,499	0.03
Scramble with bzip2	59	0.06	161	0.02	27	0.02	538	0.02	20	0.13	3,761	0.17	37,255	0.21	N/A	
DeeZ	46	0.05	187	0.03	41	0.03	452	0.02	24	0.16	3,985	0.18	35,989	0.20	2,575	0.02
DeeZ with bzip2	38	0.04	162	0.02	32	0.03	411	0.02	24	0.16	3,824	0.17	35,354	0.20	2,144	0.01
TSC	129	0.14	3,537	0.54	63	0.05	11,501	0.52	38	0.26	4,154	0.18	111,421	0.62	N/A	
Quip	47	0.05	257	0.04	72	0.06	553	0.02	25	0.17	6,029	0.27	36,337	0.20	3,767	0.02
Quip with reference	47	0.05	N/A		N/A		553	0.02	25	0.17	N/A		N/A		3,767	0.02

(d) Auxiliary fields

Table 4.8: Time and memory performance of SAM/BAM tools for 1, 4 and 8 threads. For each tool, first row shows its compression performance, while second row shows its decompression performance. Time is measured as a fraction of Samtools' performance. Memory performance is presented in MBs.

Sample	DH10B			9827.2.49			sample-2-1			K562.LID8465			dm3			NA12878.PB			HCC1954			NA12878.S1	
Threads	1	4	8	1	4	8	1	4	8	1	4	8	1	4	8	1	4	8	1	4	8	1	4
pigz	0.77	0.12	0.08	1.55	0.22	0.13	1.48	0.19	0.09	1.06	0.18	0.11	1.39	0.22	0.12	1.37	0.22	0.13	1.40	0.40	0.23	N/A	0.13
	0.63	0.26	0.26	0.82	0.46	0.46	0.49	0.24	0.24	0.70	0.71	0.69	0.79	0.57	0.56	0.70	0.84	0.86	0.91	0.73	0.95	N/A	0.60
pbzip2	1.65	0.41	0.22	1.93	0.47	0.26	4.04	0.89	0.47	3.57	0.81	0.43	0.72	0.19	0.11	0.94	0.25	0.14	1.62	0.59	0.38	N/A	0.46
	3.16	0.94	0.27	3.39	0.74	0.56	3.72	0.88	0.25	2.46	0.95	0.82	2.93	0.64	0.52	3.94	0.89	0.66	3.23	0.86	0.68	N/A	0.59
Samtools	1.00			1.00			1.00			1.00			1.00			1.00			1.00			1.00	
	1.00			1.00			1.00			1.00			1.00			1.00			1.00			1.00	
Picard	1.42			1.04			1.82			1.48			0.74			0.55			1.18			N/A	
	2.76			1.52			2.10			2.44			1.09			1.00			1.91			N/A	
Sambamba	1.05	0.12	0.09	0.93	0.15	0.11	1.12	0.18	0.15	1.05	0.16	0.14	2.48	0.25	0.13	0.93	0.14	0.09	1.39	0.32	0.22	N/A	0.13
	1.08	0.15	0.16	1.13	0.52	0.54	0.97	0.14	0.15	0.97	0.77	0.72	1.92	0.47	0.39	1.12	0.52	0.52	1.12	0.69	0.86	N/A	0.53
Cramtools (CRAM v2)	0.93			1.42			2.12			1.70			0.93			1.01			1.28			N/A	
	1.71			1.67			4.93			2.00			2.05			2.39			1.50			N/A	
Scramble (CRAM v3)	0.23	0.08	0.08	0.29	0.14	0.14	0.62	0.31	0.29	0.38	0.16	0.16	0.14	0.09	0.08	0.31	0.19	0.19	0.27	0.12	0.12	N/A	0.10
	0.76	0.18	0.19	0.66	0.49	0.51	1.58	0.78	0.83	0.67	0.65	0.63	0.58	0.48	0.48	0.84	0.73	0.64	0.71	0.42	0.43	N/A	0.50
Scramble without reference	0.29	0.08	0.09	1.18	0.12	0.12	0.45	0.18	0.21	0.43	0.16	0.16	0.21	0.10	0.09	0.46	0.22	0.20	0.37	0.11	0.11	N/A	0.10
	0.74	0.19	0.19	0.63	0.50	0.52	1.06	0.21	0.21	0.78	0.68	0.68	1.14	0.51	0.41	1.54	0.61	0.58	0.79	0.44	0.44	N/A	0.47
Scramble with bzip2	0.76	0.09	0.09	0.62	0.14	0.14	1.50	0.30	0.30	1.27	0.18	0.16	0.17	0.09	0.09	0.48	0.20	0.19	0.60	0.11	0.11	N/A	N/A
	0.89	0.19	0.18	0.66	0.61	0.58	1.72	0.74	0.75	0.81	0.78	0.77	0.67	0.41	0.40	1.63	0.54	0.54	0.82	0.43	0.43	N/A	N/A
DeeZ	0.56	0.24	0.20	0.78	0.36	0.34	1.81	0.92	0.88	0.92	0.45	0.42	0.51	0.16	0.16	0.64	0.20	0.20	0.91	0.30	0.27	N/A	0.26
	3.90	1.11	1.22	2.46	0.84	0.78	5.51	1.96	1.79	3.35	1.39	1.23	1.77	0.57	0.59	1.86	0.71	0.70	2.94	0.97	0.90	N/A	1.00
DeeZ with bzip2	0.91	0.41	0.40	1.23	0.60	0.60	3.49	1.30	1.26	2.01	0.73	0.70	0.71	0.19	0.19	1.22	0.28	0.28	1.66	0.41	0.39	N/A	0.41
	10.11	2.75	2.68	5.60	1.71	1.60	9.86	3.03	2.90	7.91	2.04	1.97	4.86	1.41	1.39	6.67	1.77	1.72	6.39	1.70	1.52	N/A	1.90
TSC	2.21			0.80			2.55			3.17			3.14			1.46			0.50			N/A	
	9.05			2.24			6.75			6.27			6.46			6.43			2.65			N/A	
Quip	0.67	0.47		0.94	0.61		1.15	0.96		1.19	0.69		0.44	0.31		0.67	0.34		0.83	0.44		N/A	0.44
	10.69	3.30		7.81	2.23		4.43	3.37		8.27	2.43		7.52	2.29		9.87	2.86		9.05	2.09		N/A	2.18
Quip with reference	0.67	0.46		N/A	N/A		N/A	N/A		1.17	0.69		0.41	0.22		N/A	N/A		N/A	N/A		N/A	0.43
	10.06	2.98		N/A	N/A		N/A	N/A		8.20	2.41		7.19	1.54		N/A	N/A		N/A	N/A		N/A	2.20
sam_comp	0.68			0.76			1.20			0.71			0.51			0.59			0.62			0.37	
	3.36			2.95			6.54			3.56			5.49			5.42			3.25			2.00	

(a) Time performance

Sample	DH10B			9827.2.49			sample-2-1			K562.LID8465			dm3			NA12878.PB			HCC1954			NA12878.S1	
Threads	1	4	8	1	4	8	1	4	8	1	4	8	1	4	8	1	4	8	1	4	8	1	4
pigz	5.80	5.80	8.00	5.60	5.80	8.20	5.80	5.90	8.10	5.80	5.90	9.90	5.60	5.80	8.20	5.60	5.80	8.90	5.80	5.70	8.10	N/A	5.90
	5.80	5.90	5.90	5.70	5.90	5.90	5.90	5.90	5.90	5.80	6.00	6.00	5.70	5.90	5.90	5.70	5.90	5.90	5.80	5.70	5.70	N/A	5.90
pbzip2	12.40	39.90	74.70	12.10	39.30	75.20	12.10	40.10	74.30	11.30	39.90	75.40	12.00	41.90	76.70	12.00	41.10	78.00	11.40	68.20	91.20	N/A	45.80
	5.80	25.20	47.90	5.70	27.20	48.60	5.90	25.70	48.80	5.80	25.10	47.50	5.70	26.20	49.40	5.70	27.10	51.40	5.80	111.80	127.00	N/A	65.50
Samtools	5.80			5.70			5.90			5.80			5.70			5.70			5.80			5.80	
	5.80			5.70			5.90			5.80			5.70			5.70			5.80			5.80	
Picard	633.50			636.60			600.80			635.00			629.40			624.40			653.20			N/A	
	591.40			694.40			1,041.00			718.00			677.50			750.30			690.00			N/A	
Sambamba	5.80	13.50	24.60	9.00	16.10	23.50	5.90	11.30	22.30	8.10	18.10	26.80	16.40	24.60	31.60	12.40	19.00	30.50	7.90	15.60	24.70	N/A	14.80
	5.80	13.30	32.10	11.10	22.00	40.40	9.00	19.00	37.60	11.30	20.20	41.80	41.00	75.70	115.60	76.50	115.40	189.10	11.30	19.40	42.40	N/A	19.70
Cramtools (CRAM v2)	2,107.20			5,109.60			4,974.30			5,332.60			12,266.60			10,342.80			11,398.90			N/A	
	6,730.50			8,227.30			10,155.70			7,641.10			10,998.70			11,526.00			11,160.00			N/A	
Scramble (CRAM v3)	39.00	309.50	288.60	279.10	1,052.90	1,418.80	288.80	1,101.60	1,577.80	582.20	810.50	1,048.90	3,102.80	6,826.70	6,704.00	2,594.70	13,517.80	12,469.70	280.20	1,042.80	1,418.00	N/A	1,143.90
	13.20	184.90	396.10	87.40	579.70	547.50	139.50	755.10	928.60	521.40	619.20	713.90	87.60	839.80	1,398.80	454.30	1,397.90	2,504.10	90.50	854.00	1,067.30	N/A	595.00
Scramble without reference	36.20	297.80	358.10	40.20	347.00	312.50	50.80	412.00	345.10	28.00	206.00	210.60	3,100.70	8,159.00	8,041.40	2,250.50	7,692.90	9,007.00	42.30	431.40	457.10	N/A	384.70
	14.70	242.10	456.20	13.60	190.40	454.70	13.40	210.10	448.50	16.00	240.90	509.80	48.60	970.50	1,800.70	77.70	1,268.40	2,065.70	17.00	432.90	858.10	N/A	275.80
Scramble with bzip2	43.50	299.60	562.80	283.20	1,049.10	1,406.60	295.80	1,183.90	1,636.30	584.30	830.20	1,099.60	3,102.80	8,261.40	8,856.00	2,596.90	17,614.50	16,426.80	286.00	1,057.90	1,450.20	N/A	N/A
	14.40	158.20	318.20	88.10	501.40	554.00	132.70	730.00	930.70	519.50	617.80	753.80	88.90	790.70	1,385.60	454.20	1,226.00	2,716.40	91.40	794.10	1,077.80	N/A	N/A
DeeZ	1,697.30	1,745.50	1,775.10	2,751.00	2,886.30	2,822.10	3,997.90	4,022.50	4,078.50	2,440.50	2,426.40	2,397.80	2,763.30	3,004.80	3,130.40	3,394.10	3,706.00	3,683.20	2,495.20	2,452.10	2,444.60	N/A	2,269.20
	3,120.70	4,793.00	6,774.60	4,004.90	5,504.80	8,444.90	3,435.40	5,011.20	7,394.20	4,303.90	5,086.70	7,736.10	4,549.20	5,466.10	6,682.80	7,683.40	8,604.30	9,711.40	3,085.60	5,113.40	8,593.10	N/A	5,642.90
DeeZ with bzip2	2,110.30	2,198.20	2,213.10	3,169.20	3,312.00	3,273.10	4,422.90	4,489.30	4,488.50	2,862.80	2,880.20	2,886.50	3,167.60	3,450.50	3,549.20	3,871.70	4,082.70	4,167.50	2,835.00	2,876.90	2,894.10	N/A	2,566.80
	3,378.00	5,255.50	6,843.50	4,411.00	5,681.90	8,334.80	3,867.20	5,746.00	7,910.30	4,725.80	5,303.20	8,258.70	5,005.50	5,856.50	7,144.10	8,210.00	9,017.00	10,183.00	3,499.90	6,055.60	8,902.10	N/A	6,391.10
TSC	14.80			16.40			14.30			1,598.30			263.90			131.60			16.30			N/A	
	16.50			13.70			18.60			1,586.10			325.20			108.60			13.70			N/A	
Quip	396.10	395.20		608.40	606.50		453.60	449.20		580.60	583.20		1,049.90	1,047.40		905.60	905.40		597.70	601.60		N/A	465.40
	395.40	395.40		600.50	600.60		447.90	449.90		576.80	576.80		982.10	979.40		792.40	793.70		593.90	594.00		N/A	459.20
Quip with reference	396.10	396.70		N/A	N/A		N/A	N/A		1,665.30	1,668.00		1,096.30	1,096.20		N/A	N/A		N/A	N/A		N/A	1,549.10
	395.70	394.00		N/A	N/A		N/A	N/A		1,662.40	1,662.50		1,031.50	1,016.50		N/A	N/A		N/A	N/A		N/A	1,544.90
sam_comp	434.40			700.50			699.60			697.50			482.40			700.60			700.20			701.10	
	436.10			698.60			699.20			698.60			482.10			699.60			700.70			699.00	

(b) Memory performance

Table 4.9: A summary of FASTQ tools' compression performance on paired-end libraries. Only those tools which support paired-end libraries were evaluated.
The left two columns indicate the overall compressed size, and the sequence-only compressed size, respectively. Right column indicates the compression and decompression times relative to Gzip (pigz), with lower being better. The last row shows the performance of sequence-only tools.

Sample	**SRR554369**		**SRR327342**		**MH0001.081026**		**SRR870667**		**ERR174310**	
Organism	*P.aeruginosa*		*S.cerevisiae*		*H.sapiens Gut*		*T.cacao*		*H.sapiens*	
Technology	Illumina GAIIx		Illumina GAII		Illumina GA		Illumina GAIIx		HiSeq	
Coverage	105×		Unknown		Unknown		65×		25×	
Original	**1,100**		**8,122**		**3,761**		**41,050**		**107,738**	
	330		*2,075*		*1,024*		*12,576*		*41,931*	
pigz	**318**	1.00	**2,229**	1.00	**1,075**	1.00	**12,308**	1.00	**36,695**	1.00
	96	1.00	*605*	1.00	*299*	1.00	*3,550*	1.00	*11,962*	1.00
pbzip2	**251**	1.19	**1,825**	1.41	**850**	1.33	**9,858**	1.07	**29,302**	0.80
	89	6.12	*550*	6.80	*279*	6.10	*3,160*	3.59	*10,950*	3.08
DSRC2	**210**	0.21	**1,505**	0.26	**698**	0.25	**8,293**	0.23	**25,988**	0.20
	83	2.15	*568*	3.22	*256*	2.06	*3,143*	1.45	*10,479*	1.25
DSRC2 (extra)	**190**	0.71	**1,333**	0.71	**637**	0.81	**7,411**	0.71	**22,784**	0.68
	78	12.01	*503*	11.48	*251*	12.71	*2,752*	6.23	*9,551*	5.96
Fqzcomp	**180**	0.35	**1,254**	0.37	**624**	0.41	**7,007**	0.34	**22,245**	0.31
	73	N/A	*445*	7.39	*240*	N/A	*2,611*	N/A	*9,252*	3.38
Fqzcomp (extra)	**188**	0.39	**1,321**	0.39	**635**	0.42	**7,377**	0.36	**22,911**	0.34
	82	N/A	*514*	7.64	*256*	N/A	*3,025*	N/A	*10,334*	3.59
Fastqz	**N/A**		**N/A**		**N/A**		**N/A**		**21,496**	3.45
									8,633	N/A
Slimfastq	**185**	0.54	**1,137**	0.48	**596**	0.54	**7,300**	0.52	**21,694**	0.47
	60	11.62	*323*	9.93	*209*	10.94	*2,365*	5.82	*8,860*	4.89
FQC	**154**	1.04	**1,105**	1.23	**604**	1.51	**6,797**	1.20	**22,382**	1.22
	N/A	12.16	N/A	13.42	N/A	18.66	N/A	6.34	N/A	5.87
SCALCE	**164**	0.59	**1,159**	0.54	**716**	0.69	**6,683**	0.57	**22,744**	0.54
	45	9.23	*221*	8.44	*198*	11.71	*1,851*	4.86	*7,548*	2.39
LW-FQZip	**236**	1.10	**1,748**	0.60	**N/A**		**8,773**	2.16	**N/A**	
	91	5.60	*639*	5.25			*3,470*	2.56		
Quip	**181**	0.49	**1,200**	0.54	**607**	0.49	**6,837**	0.50	**22,247**	0.46
	74	10.97	*393*	11.95	*229*	11.00	*2,448*	5.74	*9,114*	5.39
Leon	**176**	3.61	**1,222**	2.66	**657**	3.66	**7,822**	3.93	**26,798**	3.55
	38	16.95	*194*	17.02	*180*	14.22	*2,239*	10.49	*9,492*	9.90
KIC	**192**	5.75	**1,370**	7.40	**684**	5.32	**7,862**	6.74	**25,551**	6.19
	64	6.89	*410*	7.88	*246*	7.38	*2,674*	3.60	*9,831*	3.30
Mince		3.42		3.60		5.97		2.48		3.00
	31	2.03	*169*	1.56	*175*	2.36	*1,300*	0.72	*6,058*	0.68

4.4 Conclusion

Our evaluation of all compression tools currently available in the literature on a wide variety of data sets resulted in no overall winner that can perform well on each data type and under every performance measure we used. We conclude that an integrated solution that chooses the specific approach which performs the best on the input data type(s) with respect to the performance measure most important for the specific application would yield the best outcome, both for raw and aligned sequence data. It is interesting to note that many of the tools we benchmarked improve not only the compression rate but also the compression time of the most commonly used methods, i.e., Gzip/pigz for FASTQ files, and Samtools for SAM/BAM files. Although this is not always the case for decompression time, the time necessary for decompressing Gzip-compressed FASTQ files and Samtools-compressed SAM files, is insignificant (1/100 or less) in comparison to the most commonly used downstream analysis pipelines, e.g. for read mapping in FASTQ files, and for variant calling in SAM files, respectively. In fact, future integration of some of the best performing compression tools we tested with commonly used variant calling pipelines such as GATK (which produces multiple BAM files during execution) may significantly improve the overall running time of GATK due to smaller data footprint and thus improved locality of reference.

The decision of MPEG to issue a Call for Proposals soliciting the submission of technology for genomic data processing and storage is aimed to develop a standard compressed file format in the coming years. Such a standard will likely integrate the best features of the tools and formats evaluated in this study. The potential impact of an international standard for genomic data compression would be ground-breaking in terms of both systems interoperability and efficiency, enabling population-wide scaling of existing genomic applications.

Chapter 5

Background on ADMER Genotype Inference

Precision medicine—a concept that takes the individual variability into account when deciding treatment of the patient—has been recently reintroduced as a viable approach for development of better prevention and treatment methods [26, 144]. This has been in a large way caused by the recent advancements in high throughput sequencing (HTS) technology, which allows sequencing of patient's genome in a time-efficient and cost-effective way.

One of the first applications of precision medicine, pharmacogenomics, focuses on studying the genetic makeup of patient's genome in order to assess the patient's response to the various drugs. This study is necessary because response to a large number of clinically prescribed drugs is significantly impacted by the individual's genetic makeup [61]. For example, while some patients show a good response to a medication, the same treatment might either fail in others, or cause serious side effects, including the death of the patient [112]. It is recommended to perform accurate genotyping prior to treatment decisions that include drugs sensitive to the allelic composition of genes involved in their metabolism in order to avoid adverse effects [21]. Based on the inferred genotypes, physicians can accordingly adjust the drug selection and dosage.

Common subject of such studies are genes involved in the absorption, distribution, metabolism, excretion and response of the drugs, also known as ADMER genes. One prominent member of ADMER gene family is *CYP2D6* gene, which is involved in the metabolism of 20–25% of all clinically prescribed drugs in the human body.

Efficient computational tools are necessary in order to extract the accurate genotype information of various ADMER genes from HTS data. In most of the cases, detection of small single nucleotide variations (SNVs) is enough to properly identify the correct genotype; however, in the case of *CYP2D6* and *CYP2A6*, one needs to overcome several obstacles such are high sequence similarity, genetic recombinations with evolutionarily related pseudogenes, and high copy number variation among individuals.

In this chapter, we will: (i) provide a short overview of few important ADMER genes, and (ii) survey the commonly used ADMER genotyping platforms.

5.1 Few Examples of ADMER Genes

Many genes with observed clinical effect are categorized according to the *star-allele* nomenclature [154], which was originally devised for the genes residing in Cytochrome P450 (CYP). In such nomenclature, *1 allele is designated as a reference sequence with which polymorphic sites are compared. It should be noted that this sequence might not be the most common allele found in every ethnicity. Unique numbers (e.g. *3) are assigned to each novel allele which harbours the aminoacid substitution, or which affects the transcription, splicing or translation process. In case of non-functional nucleotide changes, additional letter is attached to the allele name (e.g. *3B). All alleles having the same unique number but different non-functional variations are called *sub-alleles* of the unique allele (e.g. *3A and *3B are both sub-alleles of *3 allele). Genes following this nomenclature are catalogued in the publicly available database maintained by Karolinska Institute [166].

5.1.1 *CYP2D6* Gene

Cytochrome P450 2D6 (*CYP2D6*) is one of the most widely studied genes for which the correlation between the allelic makeup and drug response has been established. Currently, the metabolism of 20–25% clinically prescribed drugs depends, at least in part, on *CYP2D6* genotype [81]. These include antidepressants (e.g. Prozac, Paxil, Zoloft), antipsychotics (Haldol, Risperdal), anticancer drugs (Tamoxifen), opioids (Codeine) and many others [88, 81, 74].

CYP2D6 is highly polymorphic gene for which more than 100 different alleles have been reported so far. The information about the known alleles is available at *CYP2D6* allele nomenclature website [166], which contains detailed up-to-date information about each allele. The *CYP2D6* database also includes information on the impact of genotype on the activity of the encoded enzyme for more than 50 known alleles. This list is not final, since novel alleles are regularly being found in the studies.

Based on their enzyme activity, the set of known *CYP2D6* alleles can be divided into four categories: poor (corresponding to no enzyme activity), intermediate (decreased enzyme activity), extensive (normal activity) and ultra-rapid (high activity) metabolizers [48]. Thus, the allelic configuration of *CYP2D6* might seriously affect the patient's response to the drug: for example, patients with a decreased enzyme activity will likely have the low response to pain killer codeine. In the case of codeine, ultra-rapid enzyme activity can even have fatal consequences [113]. The correlation between the *CYP2D6* phenotypes and ethnic differences is given in Table 5.1.

Table 5.1: Correlation between the *CYP2D6* enzyme activity and ethnic differences [81].

Allele	Mutation	Enzyme Activity	Allele Frequencies
*CYP2D6*2×N*	Gene duplication	Increased	Caucasians (1–5%) Black Africans (0–2%) Ethiopians (2%) Saudi Arabians (10–16%)
*CYP2D6*4*	Splicing Defect	Inactive	Caucasians (12–21%) Black Africans (1%) Ethiopians (2%) Saudi Arabians (1–4%)
*CYP2D6*5*	Gene deletion	No enzyme	Caucasians (2–7%) Black Africans (6%) Ethiopians (4%) Saudi Arabians (1–3%)
*CYP2D6*10*	Aminoacid substitution	Unstable	Caucasians (1–2%) Black Africans (51%) Ethiopians (6%) Saudi Arabians (3–9%)
*CYP2D6*17*	Aminoacid substitution	Altered affinity for substrates	Caucasians (0%) Black Africans (0%) Ethiopians (20–35%) Saudi Arabians (3–9%)

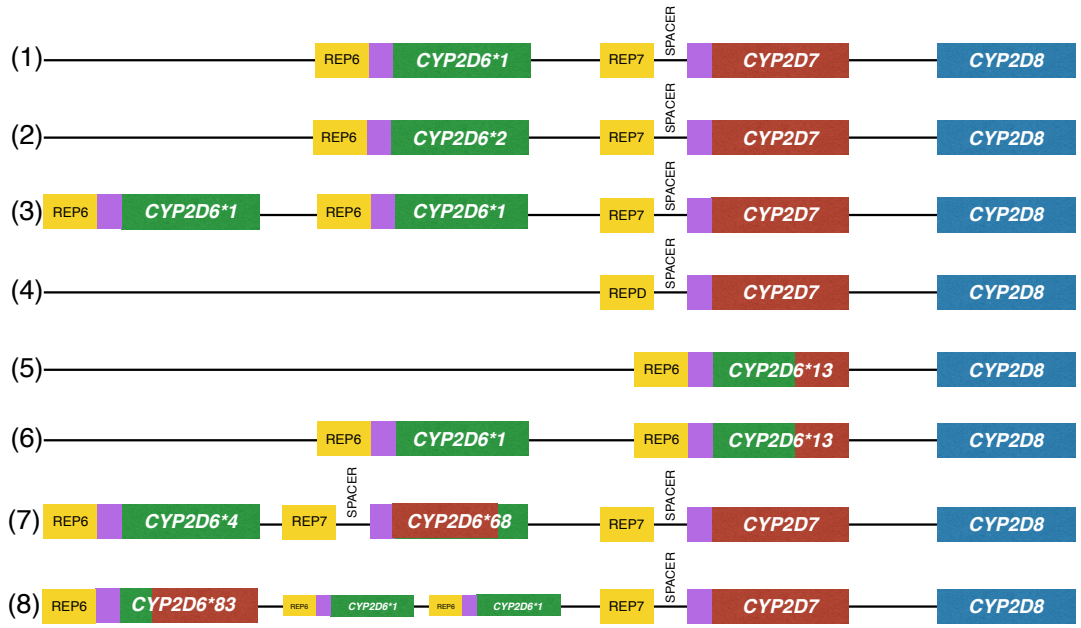
Most of the known *CYP2D6* variants are characterized by single nucleotide polymorphisms (SNPs) and short insertions or deletions (indels). However, in addition to *CYP2D6*, the human genome contains two pseudogenes *CYP2D7* and *CYP2D8*, which are evolutionarily related to *CYP2D6* [88]. The presence of highly homologous gene units in *CYP2D6* and *CYP2D7* facilitates crossovers and formation of large gene conversions, deletions, duplications and multiplications [94, 48]. Such configurations are listed in Figure 5.1.

In addition to genomic recombinations and high allelic variability, *CYP2D6* also exhibits extensive copy number variation. Although the gene might be completely absent in some cases, individuals with 14 copies have been discovered [81]. Owing to such circumstances, accurate and cost-effective *CYP2D6* genotyping largely remains a hard problem, despite its clinical significance.

5.1.2 *CYP2A6* Gene

Cytochrome P450 2A6 (*CYP2A6*) is another ADMER gene whose allelic makeup affects the metabolism of several clinically used pharmaceuticals [148]. So far, this gene has been associated with the coumarin, a naturally occurring compound present in some plants, and SM-12502, a novel platelet-activating factor receptor antagonist [148]. However, *CYP2A6* is the primarily known as the chief metabolizer of nicotine and its oxidized metabolite cotinine [70]. Around 80% of nicotine is converted to cotinine, and this reaction is largely facilitated by *CYP2A6*. Further cotinine metabolism is almost completely mediated by *CYP2A6*. Several studies have linked the *CYP2A6*'s genetic makeup with lower risk of smoking [141],

Figure 5.1: Possible *CYP2D6* configurations showing: (i) standard *CYP2D6*1* allele and related pseudogenes *CYP2D7* and *CYP2D8* (case 1); (ii) *CYP2D6* variations harbouring one or more SNVs (case 2); (iii) *CYP2D6* deletion or copy number variation (cases 3 and 4); (iv) hybrid crossovers with *CYP2D7* (case 5), and (v) various tandem arrangements between *CYP2D6* and *CYP2D7* (cases 6, 7 and 8). Image adapted from [176].



decreased cigarette consumption in adults and adolescents [161] and increased cessation [63].

Similarly to *CYP2D6*, *CYP2A6* is highly polymorphic gene with more than 40 reported variants. The *CYP2A6* variants are publicly listed in the up-to-date *CYP2A6* allele nomenclature database [166] This database also includes information about the enzyme activity for more than 20 known alleles. Enzyme activity of *CYP2A6* alleles can also be divided into four categories: none, decreased, normal and ultra-rapid activity. There is substantial variation in *CYP2A6* phenotype based on the ethnic origin, and such correlation is illustrated in Table 5.2.

While majority of *CYP2A6* alleles are characterized by SNVs, presence of highly homologous pseudogene *CYP2A7* promotes the existence of various structural variations similar to those occurring between the *CYP2D6* and *CYP2D7*. These variations include whole *CYP2A6* deletion, *CYP2A6* duplication and crossovers with *CYP2A7* [135, 148]. Typical *CYP2A6* configurations are shown in Figure 5.2. Because of structural variations and high sequence homology with *CYP2A7* (more than 95%), accurate and efficient *CYP2A6* genotyping also represents a significant challenge [181].

Table 5.2: Correlation between the *CYP2A6* enzyme activity and ethnic differences [47].

Allele	Mutation	Enzyme Activity	Allele Frequencies
*CYP2A6*1A*		Normal	Spaniards (66.5%) Japanese (42.4%) Chinese (43.2%)
*CYP2A6*1B*	*CYP2A7* gene conversion in 3' flanking region	Increased	Spaniards (30%) Japanese (37.5%) Chinese (40.6%)
*CYP2A6*2*	Aminoacid substitution	Inactive	Spaniards (3%) Japanese (0%) Chinese (0%)
*CYP2A6*3*	Gene deletion	No enzyme	Spaniards (0%) Japanese (0%) Chinese (0.7%)
*CYP2A6*4*	Gene deletion substitution	Altered affinity for substrates	Spaniards (0.5%) Japanese (20.1%) Chinese (15.1%)

5.2 Genotyping Platforms

Genotyping, or genotype inference, is the process of investigating the genetic constitution of an individual by examining the individual's DNA sequence and comparing it to the reference sequence.

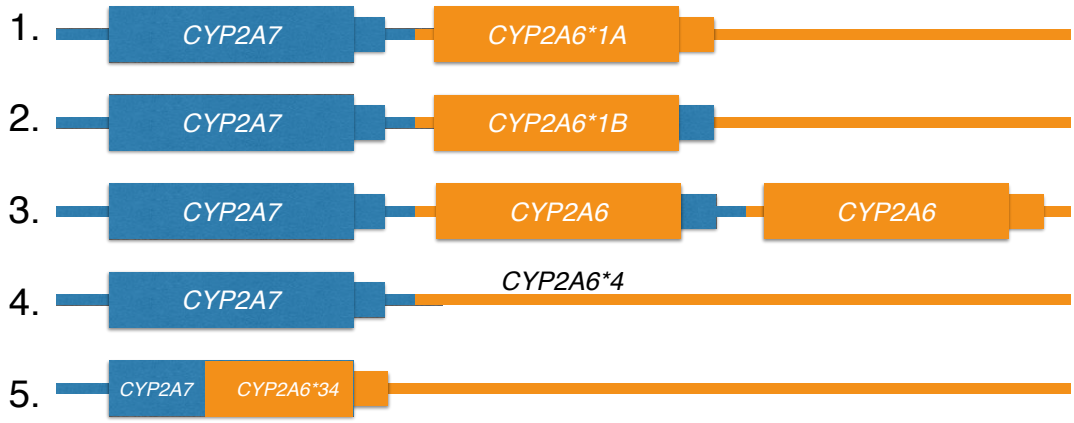
5.2.1 PCR-based Methods

Polymerase Chain Reaction (PCR) is a DNA amplifying technique [124]. It requires a DNA sample containing the target sequence of interest to be amplified, and a two small DNA sequences, called *primers*, to be attached to the target. The target DNA helix is first split into the two strands, and PCR primers are attached to the newly formed strands. Afterwards, the remainder of each strand is completed with matching dNTPs (*deoxynucleoside triphosphates*, or blocks containing one of the nucleotides) with the help of appropriate enzyme (e.g. Taq polymerase). This process can be continued as long as necessary, wherein the amount of DNA is doubled at each step. The final result of this process is the amplification of the DNA sequence falling between the primers.

The following genotyping methods use PCR in order to identify various SNPs of interest:

Amplification refractory mutation system PCR (ARMS-PCR) [129] uses primers specifically designed to overlap at the target SNP location. If they do not match the SNP (meaning that SNP is not present), nucleotides with a mismatched 3'-residue will not function as primers in the PCR under the appropriate conditions. As a result, the ARMS-PCR will produce desired results only if the primers attach themselves to the SNP for which they were designed for.

Figure 5.2: Possible *CYP2A6* configurations showing: (i) standard *CYP2A6*1* allele (orange) and related pseudogene *CYP2A7* (blue); (ii) *CYP2A7* conservation in 3' flanking region of *CYP2A6*; (iii) *CYP2A6* gene duplication; (iv) *CYP2A6* deletion (denoted as *CYP2A6*4*); and (v) hybrid crossover with *CYP2A7*. Image adapted from [148].



Primer extension approach relies on the primers that attach themselves to the target DNA immediately upstream of the SNP nucleotide (i.e. a nucleotide next to the primer's end will be a SNP nucleotide). Afterwards, special fluorescently labelled dNTPs (or dNTPs with other kind of detectable signal) will attach themselves to the SNP nucleotide [62]. At the end of the process, allelic configuration of the sample is estimated from the amount of remaining fluorescent labels usually by the application of mass spectrometry. Primer extension is suitable for high throughput analysis, and many SNPs can be genotyped at once by this technology.

Commercial technologies which rely on primer extension include SNPstream assays [11], iPLEX ADME PGx panels and Illumina's Infinitum assays. While these technologies provide high sample throughput and low operating costs, they come with low flexibility for incorporation of new assays [175].

TaqMan assays [71] use special primers which can amplify the region containing the targeted SNP. In addition to these primers, TaqMan assays come with the allele-specific probes, containing a fluorophore and quencher molecules attached to their 5' and 3' ends, respectively. If the allele-specific probe links to the SNP (thus confirming the allele's existence), it will get degraded by Taq polymerase, an agent which is used to link the dNTPs to the DNA strand. This degradation will result in fluorophore's separation from quencher molecule, which will generate a noticeable signal. TaqMan probes can also be used for multiple SNP detections in a single run, as long as SNPs are far enough from each other (since separate allele-specific probe is required for each SNP).

TaqMan ADME assays are widely used for SNP genotyping, mostly due to their simplicity and high sample throughput. However, TaqMan assays are often inconsistent, and they cannot detect various structural variations, including the copy number change [175, 146, 143, 45, 151].

DNA Microarrays

DNA microarrays are collection of small DNA primers containing various SNVs attached to the single chip. Typically, a microarray chip contains a set of primers describing all allelic configurations of the targeted gene. Chips can be designed to target multiple genes at once. Initially, the messenger RNA (mRNA) is extracted from the sample being analyzed. It is then coloured, converted to the cDNA, and attached to the chip. cDNA will bind to the primers which are matching its nucleotides, and each binding will be marked with cDNA's colour. Unbound cDNA fragments will eventually get discarded. Finally, the picture of chip is taken and processed via image processing software in order to detect the allelic composition of the sample [98].

Typical examples of microarray-based assays include Affymetrix DMETPlus arrays, Roche AmpliChip CYP450 and Illumina VeraCode ADME panels. However, these technologies cannot efficiently detect copy number variations [175].

It should be noted that all of the abovementioned methods require in advance knowledge of the allele-specific SNVs, since special primers need to be constructed for each queried SNV.

5.2.2 Sequencing-based Methods

Sanger sequencing [158], the current Food and Drug Administration's (FDA) gold standard for sequencing [175], is capable of producing long reads (in range of 650–800bp) with reasonable error rates. However, such approach is usually too costly, labour-intensive and time-consuming. Additionally, Sanger sequencing is not able to adequately determine large indels or copy number variants [109, 48, 175].

HTS whole genome (WGS) and whole exome (WES) sequencing provide the most comprehensive method for obtaining the individual's genetic variation. While WGS platforms are able to capture the whole individual's genome, WES only covers the expressed genes in the genome. Both WGS and WES provide high throughput and high coverage of the data. Additionally, WGS can be used to detect the novel SNVs and previously unknown allelic variants, thanks to its low error rates. Nevertheless, short read lengths and platform-specific biases [35] introduce significant challenges in the interpretation of WGS data. Furthermore, many existing WES methods do not cover non-coding regions, which makes WES somewhat impractical for pharmacogenomical purposes [111].

For the reasons described above, cheaper (but more error-prone) PCR and microarray based methods are still the most popular technologies for genotype inference of ADMER genes (and *CYP2D6* in particular) [48].

PGRNseq sequencing [59] is the recently introduced custom-capture panel designed to sequence 84 ADMER genes in a HTS fashion. This sequencing technology uses Illumina HiSeq 2000 platform for sequencing, and produces reads covering around 968 kilobases of the genome with the average depth of coverage of 500×. Currently, the cost of PGRNseq is ten times lower than WGS and two to three times lower than WES. However, PGRNseq is still prone to the data interpretation ambiguities, particularly with the presence of structural variants in the genes like *CYP2D6* and *CYP2A6*.

Detailed information about the various genotyping methods is available in the literature [146, 175, 98].

5.3 Computational HTS Genotyping Methods

The high coverage of HTS data, together with its low cost, offers a feasible genotyping strategy for many pharmaceutical genes. Here we will present a few common techniques which are used to infer genotypes from HTS data.

The first step in the genotype analysis consists of short read alignment to the reference genome. This is typically done with aligners like BWA [106], Bowtie [100, 99] or mrsfast [65, 68]. After the alignment, a simple but crude approach for SNP calling would consist of analyzing the aligned bases at the SNP loci as follows. All bases with low sequencing or mapping quality are discarded, assuming that they represent the sequencing artefacts. If the percentage of one base goes below the predefined threshold, the evidence supporting that base is discarded (e.g. if we have 50 bases supporting G and only 2 bases supporting C at the SNP loci, chances are that C is the result of sequencing error; thus, all bases containing C are discarded). Finally, all locations which still show the evidence of variation are called as SNPs.

Probabilistic modelling

A more sophisticated way of genotyping uses Bayesian likelihoods to estimate the genotypes [108]. Such estimator computes the posterior probability $P(G \mid D)$ for each genotype G, and selects the genotype which maximizes the posterior. Posterior is calculated as:

$$P(G \mid D) = \frac{P(G)P(D \mid G)}{P(D)},$$

where D denotes the input data. Since $P(D)$ is constant for given D, it is effectively ignored in the calculation. $P(G)$ is prior probability of genotype G, and it is influenced by zygosity

of the reference, as well as the correlation between the individual being analyzed and sample in which the individual belongs. In a naïve implementation, if we assume that genotype G consists of the allele having a single SNP taking values A_1 and A_2 at loci l, $P(D \mid G)$ can be estimated as:

$$P(D \mid G) = \prod_{b \in \text{pileup}(l)} P(b \mid G),$$

where

$$P(b \mid G) = \frac{1}{2} P(b \mid A_1) + \frac{1}{2} P(b \mid A_2).$$

In this estimation, pileup(l) represents the set of bases covering the loci l from the reads which map over l. Naïve model can estimate $P(b \mid A)$ as $e(b)/3$ if $b \neq A$, or as $1 - e(b)$ if $b = A$, where $e(b)$ describes the sequencing error rate of the nucleotide b (usually obtained from the quality score).

Samtools [107, 105] assumes data independence across the different loci. It also assumes that all variants in the sample are biallelic. The estimator for likelihood $L(G)$ of the genotype G is roughly given as:

$$\begin{aligned} L(G) = & \frac{1}{m^k} \times \prod_{b=A_1} \left[(m - P(G))\, e(b) + P(G)\, (1 - e(b)) \right] \\ & \times \prod_{b=A_2} \left[(m - P(G))\, (1 - e(b)) + P(G) e(b) \right], \end{aligned}$$

where m denotes the ploidy of the sample (usually 2 for haploid samples) and k denotes the number of bases covering the allelic site.

In practice, Samtools uses a slightly modified version of the abovementioned equation which also takes into the account various error dependencies [105].

Genome Analysis Toolkit (GATK) [120, 32] is a full-scale MapReduce framework for HTS data analysis. One of the framework's key components is the variant caller and genotype estimator.

Before genotype estimation is conducted, GATK applies the following filtering and alignment improving steps:

1. **Local realignment**. Due to the common misalignments occurring in the presence of indels, GATK performs a multiple sequence realignment around the potential indel sites in order to refine the alignments.

2. **Duplicate marking**. GATK tries to eliminate all of the molecular or PCR duplicates from the mapping.

3. **Base quality recalibration**. In this step, base qualities (i.e. error rates $e(b)$) are corrected based on the neighbouring quality scores and the sequencing-specific information (e.g. machine cycle, tile information etc.).

Genotype is estimated via expectation-maximization (EM) algorithm which uses the similar likelihood calculation as in Samtools [104].

Constellation [176] is a *CYP2D6* genotyping framework which is able to report both genotypes and phenotypes described in the *CYP2D6* database [166]. It uses GSNAP [186] and GATK [120] for initial mapping, read realignment and SNV calling. Afterwards, it constructs the list of all possible diplotypes D_i occurring in *CYP2D* locus, and calculates the similarity score between the output of GATK SNP caller, V, and each diplotype D_i as follows.

Let $X = V \cap D_i$ be the set of variants shared by V and D_i, $Y = V \setminus D_i$ a set of variants unique to V, and $Z = D_i \setminus V$ a set of variants unique to D_i. Let **pred** denote the event when the variant is predicted, and **pres** the event when the variant is present. Define sensitivity of the variant as

$$\mathbf{sens} = P(\mathbf{pred} \mid \mathbf{pres}),$$

and specificity as

$$\mathbf{spec} = P(\neg\mathbf{pred} \mid \neg\mathbf{pres}).$$

The final score determining the likelihood of D_i as genotype is given as:

$$\text{score}(D_i) = \left(\frac{\mathbf{sens}}{1-\mathbf{spec}}\right)^{|X|} \times \left(\frac{1-\mathbf{sens}}{\mathbf{spec}}\right)^{|Y|} \times \left(\frac{1-\mathbf{spec}}{\mathbf{sens}}\right)^{|Z|},$$

where $|X|$, $|Y|$ and $|Z|$ denote the cardinality of the sets X, Y and Z, respectively. Finally, D_i with the highest score is selected as the sample's genotype.

At the time of writing, Constellation is not publicly available for download.

5.4 Conclusion

In this chapter, we presented an overview of the few pharmaceutically important genes, namely *CYP2D6* and *CYP2A6*, whole allelic makeup can impact the drug treatment decisions. Afterwards, we listed some common techniques and platforms used for genotyping of such genes.

Genotyping can be done either by the use of specialized PCR or microarray based assays, or by sequencing of the genomic region of interest. While specialized assays offer fast and cost-effective genotyping, they are limited to the specific set of genes and variations, produce inconsistent results, and are not able to deal with various structural variations involving the target genes. Sequencing methods provide a possible solution to those issues, but require

novel computational methods to be developed in order to accurately interpret the often ambiguous sequencing data.

Chapter 6

Exact genotyping of *CYP2D6* gene using high throughput sequencing data

Response to a large number of clinically prescribed drugs varies significantly among individuals. While some patients show a good response to a medication, the same treatment might fail in others or cause serious side effects which can even result in the death of the patient [112]. In many cases an individual's genetic makeup has been recognized as one of the potential causes of treatment failures [61]. In order to avoid adverse effects, it is recommended to perform accurate genotyping prior to treatment decisions that include drugs sensitive to the allelic composition of genes involved in their metabolism [21]. Drug dosage and selection can then be adjusted based on the inferred genotypes.

Cytochrome P450 2D6 (*CYP2D6*) is one of the most widely studied genes for which the correlation between the allelic makeup and therapy response has been established. It is currently estimated that metabolism of 20–25% of clinically prescribed drugs is, at least in part, dependent on *CYP2D6* genotype [81]. These include antidepressants, antipsychotics, anticancer drugs, opioids and many others [191, 81].

CYP2D6 is highly polymorphic gene with more than 100 different allelic variants reported up to date. The information about the known alleles is publicly available at *CYP2D6* allele nomenclature website (`http://www.cypalleles.ki.se/cyp2d6.htm`) which contains detailed information on the sequence variants characterizing each allele. The website also includes information on the impact of genotype on the activity of the encoded enzyme for more than 50 known alleles. Based on their enzyme activity, the set of known *CYP2D6* alleles is divided into four categories: poor (PM), intermediate (IM), extensive (EM) and ultra-rapid (UM) metabolizers corresponding to "none", "decreased", "normal" and "ultra-rapid" activity, respectively [50]. As genotyping techniques improve and more studies including large cohorts of individuals with different ethnic backgrounds are conducted, the existing

database will expand to include novel alleles and more detailed, more accurate information on genotype-phenotype associations for known alleles.

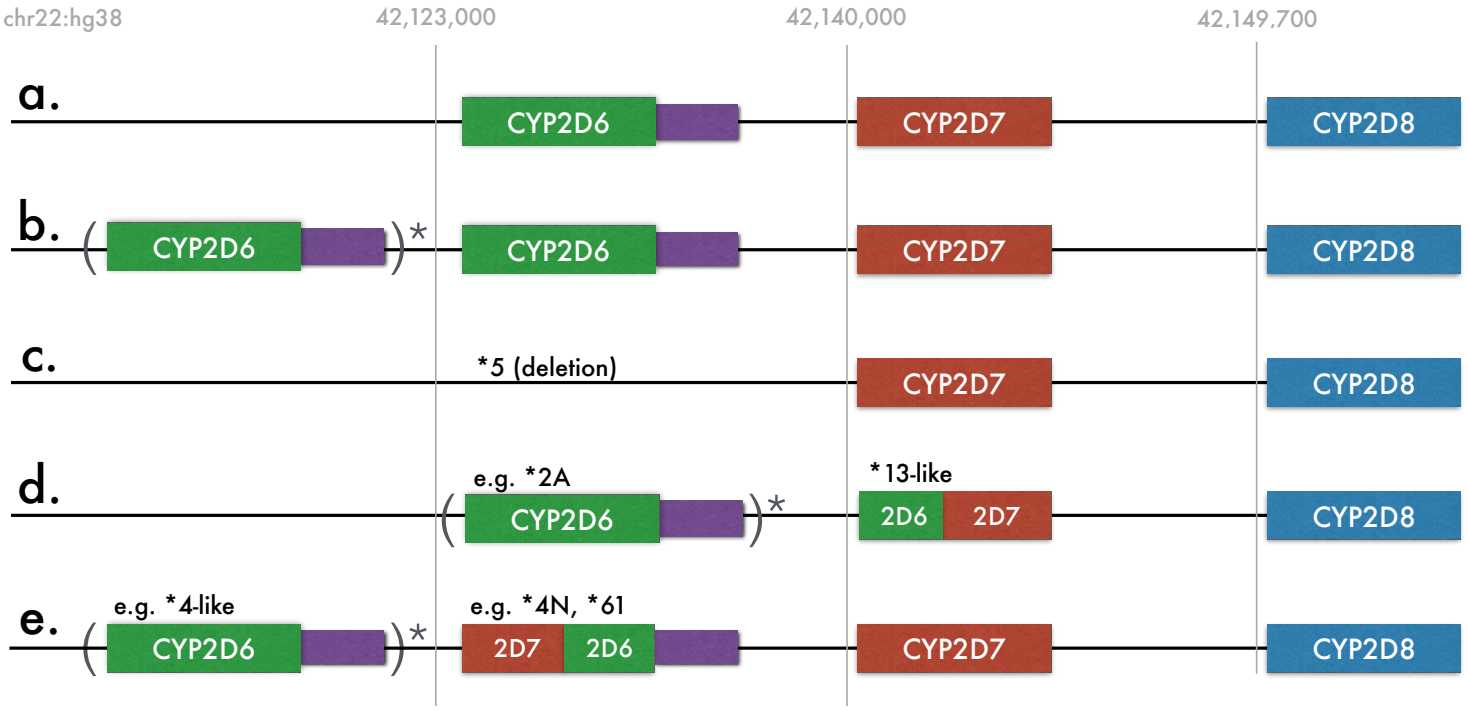
Most of the known *CYP2D6* variants are characterized by single nucleotide polymorphisms and short insertions/deletions (indels). However, in addition to *CYP2D6*, the human *CYP2D* locus contains two pseudogenes *CYP2D7* and *CYP2D8*, closely located and evolutionarily related to *CYP2D6* [88]. The presence of highly homologous gene units in *CYP2D6* and *CYP2D7* facilitates crossing-over and formation of large gene conversions, deletions, duplications and multiplications [94]. Figure 6.1 depicts all of the known *CYP2D* gene arrangements.

CYP2D6 also exhibits extensive copy number variation. While the gene might be completely absent in some individuals, others who carry as many as 14 copies have been discovered [81].

Due to its clinical significance and the prevalence of genotypes resulting in altered phenotypes, several *CYP2D6* genotyping platforms have been introduced. These usually include allele-specific primer extension assays, liquid bead arrays and TaqMan genotyping assays. However, several discrepancies among genotypes produced by these platforms have been reported [143, 45]. Also, discoveries of some of the novel alleles and variations necessitate the extension of existing kits by construction and addition of novel primers thereby increasing the time and cost required for genotype inference. The sensitivity of primers to sequence variation in primer binding sites can result in incorrect genotype assignment [49]. Furthermore, some of the techniques are incapable of detecting several alleles [45]; they can also produce ambiguous readouts or incorrect estimates for individuals carrying hybrid genes [49, 94]. Another issue with some of the available approaches is their inability to differentiate between duplicated and non-duplicated alleles in samples with a duplication signal and heterozygosity [94].

Recently introduced high throughput sequencing (HTS) technologies represent a promising, time-efficient, cost-effective and potentially high-accuracy alternative to currently used genotyping techniques. In a single machine run, a typical HTS sequencing platform, like Illumina HiSeq 2000, generates billions of short DNA fragments/reads. Although these reads are substantially shorter than those generated by Sanger sequencing (75–250bp vs 650–800bp), their higher coverage provides improved indel and SNP detection accuracy. In addition, because leading HTS platforms (in particular Illumina) provide uniform sequence coverage, the copy number of a genomic region of interest can be estimated by comparing the expected and observed coverage in a given genomic region. Furthermore, the use of paired-end reads can facilitate fine-grained inference of the origin of sequence variants commonly observed in both *CYP2D6* and *CYP2D7*.

Figure 6.1: Five known *CYP2D6* gene arrangements. The reference strand of human genome was used in all cases. Various number, including zero, of *CYP2D6* copies is allowed within the parenthesis. (a) *CYP2D6* non-duplicated arrangement consisting one copy of each of *CYP2D6*, *CYP2D7* and *CYP2D8*. Purple rectangle represents *CYP2D6* untranslated region. This region contains several variations important for the detection of some *CYP2D6* alleles; (b) typical *CYP2D6* duplication arrangement; (c) the deletion arrangement, indicating the absence of *CYP2D6* (denoted as *5 allele); (d) *CYP2D6/2D7* fusions (*13 family of alleles) lacking *CYP2D7*. Variable number of copies of *CYP2D6* gene might precede fusion alleles; (e) *CYP2D7/2D6* fusion cases with presence of *CYP2D7*. Variable number of copies of *CYP2D6* gene might precede fusion alleles in this case as well.



Despite rapid advances in HTS technologies, no available computational tool is capable of resolving the *CYP2D6* genotype. A computational tool to solve this important problem needs to address many obstacles emerging from extensive allelic variation and sequence similarity between genes present at the *CYP2D* locus. Although this locus is unique in the human genome, the high degree of similarity among *CYP2D* genes results in an abundance of reads with multiple mapping locations. This can significantly complicate copy number analysis and accurate genotyping. As a large number of SNPs and indels that define some of *CYP2D6* alleles can also occur in the pseudogene *CYP2D7*, detailed analysis of obtained variation signals is necessary. Failing to do so might result in inaccurate *CYP2D6* genotype assignment caused by improper interpretation of variations originating from *CYP2D7*, mistakenly assigned to *CYP2D6*.

In this work, we present Cypiripi, the first algorithm for automatic *CYP2D6* genotype inference from genomic HTS data. Cypiripi is able to properly resolve complicated configurations, including fusions between *CYP2D6* and *CYP2D7* genes, as well as both *CYP2D6* and *CYP2D7* deletions and duplications. We demonstrate that Cypiripi is highly accurate through extensive experiments involving both simulated and real datasets.

6.1 Methods

Cypiripi consists of the following main steps (Figure 6.2):

1. Library preparation step, where a library containing the complete set of relevant variations occurring in *CYP2D* locus is constructed;

2. Read alignment step, where each HTS read is aligned to the library of gene variants from the *CYP2D* locus determined in the library preparation step;

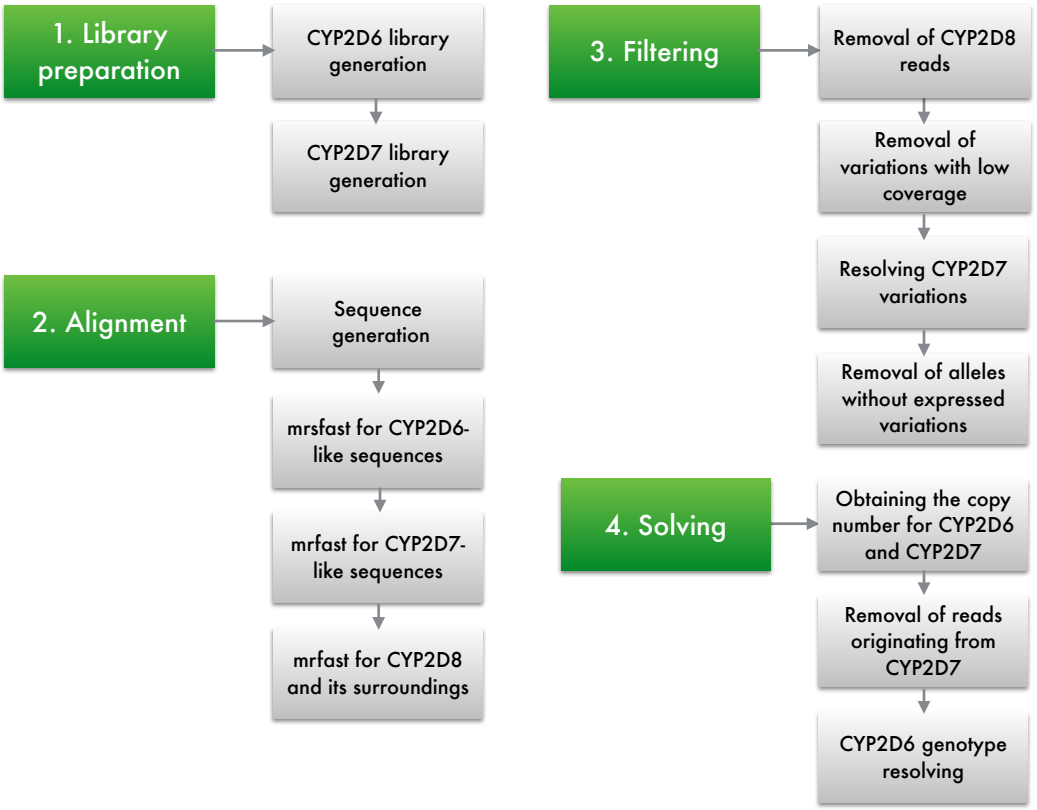
3. Filtering step, where alleles with sequence variations that lack appropriate read support are removed from further consideration;

4. Combinatorial optimization step, where the genotype, consisting of the composition of *CYP2D6* allelic variants and their copy numbers, is inferred by using Integer Linear Programming (ILP).

6.1.1 Library preparation

In this step, we construct a library containing the information about currently available variations occurring within *CYP2D* locus. These include SNPs, indels and details about recombination events occurring between *CYP2D6* and *CYP2D7*. Due to the nature of the problem being solved, we mainly focus on variations that define the currently known *CYP2D6* alleles.

Figure 6.2: Graphical representation of the steps employed by our framework.



Variations occurring in *CYP2D6* have been extracted from the most recent update (December 2014) of the database at the *CYP2D6* allele nomenclature website. The corresponding information is stored in the simple text file and any subsequent changes in online database can be easily incorporated in our tool by a straightforward modification of this file.

CYP2D7 library reconstruction is harder due to the fact that there is no basepair level characterization available for *CYP2D7* alleles. In order to be able to differentiate *CYP2D7* from *CYP2D6*, we used 10 available *CYP2D7* sequences from GenBank and other sources:

1. M33387
   (http://www.ncbi.nlm.nih.gov/nuccore/M33387)
2. NW_003315971.2
   (http://www.ncbi.nlm.nih.gov/nuccore/NW_003315971.2)
3. NT_187682.1
   (http://www.ncbi.nlm.nih.gov/nuccore/NT_187682.1)

4. NC_000022.11
   (http://www.ncbi.nlm.nih.gov/nuccore/NC_000022.11)
5. AC_000154.1
   (http://www.ncbi.nlm.nih.gov/nuccore/AC_000154.1)
6. NC_018933.2
   (http://www.ncbi.nlm.nih.gov/nuccore/NC_018933.2)
7. ENSG00000205702.2
   (http://www.ensembl.org/Homo_sapiens/Gene/Summary?g=ENSG00000205702)
8. hg19 reference genome
   (chr22:42536214–42540575)
9. hg38 reference genome
   (chr22:42140203–42144549)
10. NA12878 Assembly, Maternal Chromosome
    (22:42534697–42539033)
11. NA12878 Assembly, Paternal Chromosome
    (22:42534225–42538562).

These sequences were aligned with Clustal [101] in order to obtain a consensus alignment for *CYP2D7* gene. This consensus was aligned to the *CYP2D6* reference allele, and the set of differences between those two consensus sequences were used as markers for identifying *CYP2D7* presence, and for generating *CYP2D7* reference sequence. These markers were also used for proper description of the fusion and conservation alleles (i.e. alleles involving a portion—e.g. a whole exon—of a *CYP2D6* allele, swapped with the similar sequence portion of a *CYP2D7* allele). Note that those markers are not intended to be authoritative, since there might exist uncatalogued *CYP2D7* alleles which do not contain any of those markers. Our formulation takes that into consideration, and uses only markers which have sufficient support to infer the presence of *CYP2D7* (e.g. insertion of T at loci 137 indicates with high probability that *CYP2D7* region is present).

Our method does not require a database with *CYP2D8* variants for the reasons described below.

6.1.2 Read alignment

We established the uniqueness of *CYP2D* gene sequences by searching for the entire human genome (excluding the *CYP2D* locus) regions with high sequence similarity to the *CYP2D* genes by BLAT [87]. No subsequence of length > 60bp from any one of the *CYP2D* genes can be found in the remainder of the human genome within an edit distance of 6.

As the length of the reads generated by current HTS technologies usually exceeds 60bp, each of the reads that can be aligned (with only a few differences) to *CYP2D6* genes must originate from the *CYP2D* cluster. Thus, in order to extract the reads originating from *CYP2D* genes it is sufficient to map all of the reads against this set and discard those that cannot be aligned successfully. This alignment is performed by our in-house developed mrfast and mrsfast family of multi-mapping tools [4, 65, 68]. For each of the successfully aligned reads, we keep the details about the genes and locations it can be aligned to, and variants in the library it supports. Alignment details respectively for *CYP2D6*, *CYP2D7* and *CYP2D8* are given below.

In order to perform an alignment of the reads against all possible *CYP2D6* alleles, we constructed each allele at basepair resolution. For that we combined information from *CYP2D6* reference sequence M33388 (http://www.ncbi.nlm.nih.gov/nuccore/181303) and *CYP2D6* variant database we constructed in the previous step. The alignment was then performed using mrsfast, allowing at most 2 mismatches per read.

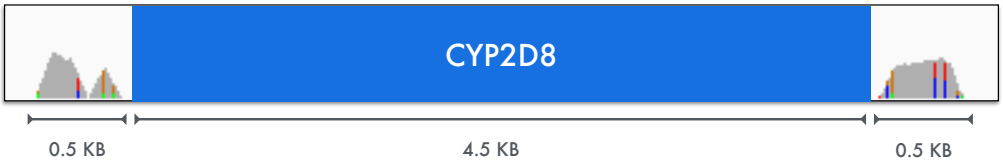
Due to the lack of a comprehensive list of sequence variants commonly observed in *CYP2D7*, there could be SNVs or short indels not represented in the consensus sequence we constructed. In order to account for such variants and possible sequencing errors we set the maximum number of errors (mismatches and indels—i.e. edit distance) to 5. The alignment of reads was performed by mrfast, which allows mismatches as well as indels. Since reads originating outside of the *CYP2D* locus have an edit distance more than 5 to any *CYP2D* gene, any read aligning with the consensus sequence should be originating from the *CYP2D* locus.

Although *CYP2D8* is evolutionarily related to *CYP2D6* and *CYP2D7*, its sequence composition is significantly different from that of the other two. In addition, there are no recombination events involving *CYP2D6* and *CYP2D8*. As a result, we assume that *CYP2D8* is always located downstream of *CYP2D7* (considering 5' to 3' orientation in the human reference genome) and that, the vast majority of the reads originating from this gene are not mappable to the other two genes. However, there are two 0.5kb flanking regions at *CYP2D8* boundaries that can give rise to some reads mappable to *CYP2D6* and/or *CYP2D7* (Figure 6.3). Such reads can interfere with copy number and other key estimates for *CYP2D6* and *CYP2D7*. It is therefore important to filter out such reads. We thus perform an alignment of the reads against *CYP2D8* gene and its surroundings by using the *CYP2D8P* reference sequence M33387 (http://www.ncbi.nlm.nih.gov/nuccore/M33387) by the use of mrfast, with edit distance 5. Reads falling into the abovementioned flanking regions around *CYP2D8* are marked for filtering, as described in the next step.

6.1.3 Filtering

After the first two steps we obtain a set of reads mappable to at least one of the *CYP2D* genes together with the details about the exact locations they align to and the sequence

Figure 6.3: The coverage of the reads mappable to the *CYP2D6* and/or *CYP2D7* genes is depicted in grey on the flanking regions of *CYP2D8* (blue strip). Only two small 0.5 KB regions on the sides *accept* *CYP2D6* and/or *CYP2D7* reads.



variants they support. In addition, for each variant we have information about the number of reads supporting it. In order to remove false positives and lower the search space for the combinatorial optimization step, we perform several read, variation and allele filtering steps with the details below.

***CYP2D8* read filtering:** Since we know the regions in the *CYP2D6* and *CYP2D7* to which some of *CYP2D8* reads can map to, we can use the surroundings of these regions to find out the excess coverage generated by such reads. After finding excess coverage, we remove the reads in order to "flatten" the coverage of the region with its surroundings.

Variation filtering: Sequencing errors can result in support for sequence variants that do not exist in the underlying genome. For example, assume that our library contains the SNP G>A at genomic position p. Also, assume that the underlying genome does not contain this SNP. In principle, a sequencing error occurring at position p may result in some support for nucleotide A instead of G. Due to the low sequencing error in the data we use, it is very unlikely that read support for non-existing variants will be significant. We therefore filter out all potential sequence variants that have support lower than user-specified parameter η.

***CYP2D7* variation filtering:** It is of great importance for our method to detect all variation coming from the *CYP2D7* reads which are falsely aligned to some of the *CYP2D6* alleles. This is particularly important due to the fact that many key sequence variants used for *CYP2D6* allele identification might also occur in *CYP2D7*. For example, c.1661 G>C is commonly found both in *CYP2D7* and many of the *CYP2D6* alleles. It is usually not clear whether this variant is associated with *CYP2D7* or *CYP2D6* or both. Commonly, *CYP2D7*-specific variants are found within the close vicinity (usually within the 100bp) of the shared variants. This helps with the detection of the origin of shared variations by using only read alignment information. Unfortunately, in some cases, *CYP2D7*-specific variants are not present in the vicinity of shared variants (within a distance comparable to the read length). Unresolved shared variants can falsely indicate the existence of specific *CYP2D6*

alleles, which, in reality may not be present in the sequenced genome. For example, c.3853 G>A is shared by both *CYP2D6*27* and *CYP2D7*; the closest *CYP2D7*-specific variation is more than 100bp away from this locus. Relying solely on this information, we cannot decide whether it is *27, or, *1 (the wild type allele) together with a *CYP2D7* harbouring this variation, that is present. Fortunately, this problem is resolved through the use of paired-end sequencing with a fragment length of 300bp or more, whose span would help detect *CYP2D7*-specific variants.

Allele filtering: About 60% of the *CYP2D6* alleles from the database are easily distinguishable from other alleles by at least one unique variant in their characterization. Each *CYP2D6* allele whose unique variants are not supported after previous filtering steps are removed from further consideration. Unfortunately, the absence of a comprehensive list of *CYP2D7* variants prevents us from applying this stringent filtration rule to the *CYP2D7* gene.

6.1.4 Combinatorial optimization

The goal of the combinatorial optimization step is to find a genotype which best describes the set of reads remaining after previous read filtering steps. The optimal genotype is supposed to match the observed read coverage as closely as possible, as explained below.

Notation

Let L denote the set of variants from *CYP2D6* and *CYP2D7* variation library that have non-zero read support after previous filtering steps.

Consider an arbitrary variant $w \in L$. Assume that w starts at position j in the *CYP2D6* reference sequence. In addition to the reads supporting w, there might also exist some reads spanning location j and not supporting any variation from L at location j. Since our optimization step also requires the number of such reads to be available, in order to detect the wildtype (*1) and other alleles without any variation at location j, we introduce the notion of a *neutral SNP variation* denoted by $n(w)$, defined as the special type of "variation" that preserves the reference nucleotide at location j. To illustrate this, consider the following example where w denotes the c.1661 G>C in *CYP2D6*. Since this SNP starts at position 1661, the corresponding $n(w)$ in this case is defined to be c.1661 G>G. Neutral SNP variation $n(w)$ is harboured by all alleles that do not harbour any variation starting at j.

Now we define a set V of all variants (including neutral SNPs) as:

$$V = \bigcup_{w \in L} w \cup \bigcup_{w \in L} n(w).$$

Let *coverage* of $v \in V$ be the number of unfiltered reads supporting v, and be denoted by $\text{cov}(v)$.

Let V_i denote the set of variations defining the i-th allele. Clearly $V_i \subseteq V$.

Formulation

The problem is formulated as an instance of integer linear programming (ILP), and solved using IBM CPLEX optimization software. In order to formulate the problem, we use the assumption that the average coverage of the HTS experiment is uniform, and that its value is the user-provided parameter λ.

Define a_i as an integer variable denoting the number of copies of the i-th *CYP2D6* allele in the given sample. Let $\mathbf{a} = (a_1, \ldots, a_N)$, where N denotes the number of different alleles. We assume that the total number of copies of *CYP2D6* is upper bounded by a given parameter c. In this study we set $c = 20$ which is greater than the maximum number of *CYP2D* copies found so far in a single individual (see Chapter 6 introduction). In order to incorporate this into our ILP we add the constraint $0 \leq a_i \leq 20$ for each i.

The expected coverage of variation v_j is given as a function of $\mathbf{a}$ and λ as follows:

$$\lambda \sum_i \delta_{ji} \cdot a_i.$$

where $\delta_{ji} = 1$ if $v_j \in V_i$. Otherwise, we set $\delta_{ji} = 0$.

The difference between the expected and obtained coverage for variation v_j, denoted as e_j, is then given by:

$$e_j = \text{cov}(j) - \lambda \sum_i \delta_{ji} \cdot a_i.$$

The value of each e_j is bounded by ϵ, where ϵ is user-defined parameter.

$$|e_j| \leq \epsilon \qquad \text{for each } j \tag{6.1}$$

Our goal is to set the values for a_i so that the sum of absolute values of all e_i is minimized. Thus, we define our objective function as:

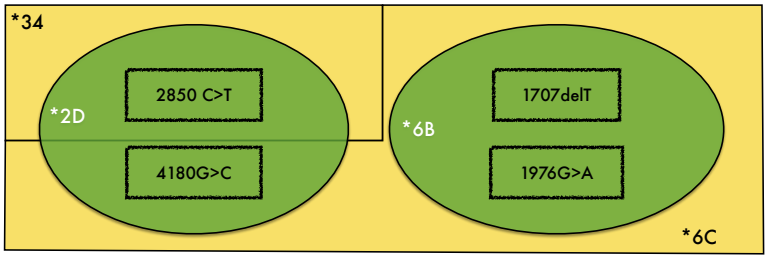
$$\min_{\mathbf{a}} \sum_j |e_j|. \tag{6.2}$$

Note that the performance of this model highly depends on the accuracy of *CYP2D6* database, in particular the choice of V and V_i for any i. For this reason, any allele which is not present in the database will be assigned a closest allele from the database in the optimal solution.

We use a two-stage approach to solve the genotyping problem. In the first stage, we use previously described ILP formulation to obtain the copy number for *CYP2D6* and

Figure 6.4: The ambiguous case where two genotypes *6C/*34 (yellow) and *2D/*6B (green) are equally likely. Key c.2850 C>T and c.4180 G>C are too distant to be resolved with the currently available HTS data.

$$A_{34} \cup A_{6C} \equiv A_{2D} \cup A_{6B}$$



CYP2D7, without making any decision about *CYP2D6* genotypes. Based on this, we can remove the *CYP2D7* reads and estimate the exact coverage at each location. Since the removal of *CYP2D7* reads also removes the support for many shared variations, we perform an additional round of filtering in order to further reduce the number of potential false positives. Finally, we invoke a slightly modified version of the abovementioned ILP formulation in order to detect specific *CYP2D6* genotypes, as described below.

In the end, we are only left with the reads (assumed to be) originating from the *CYP2D6* gene. Assume that at this stage some $v \in V$ has non-zero coverage. Denote by A the set of *CYP2D6* alleles harbouring v. The existence of reads supporting v is now a clear indication for the existence of at least one allele from the set A in the sample being analyzed. Thus, in addition to the above constraints, we add the following constraint for each v that has non-zero coverage:

$$\sum_{i \in A} a_i \geq 1.$$

After the optimal solution for this extended ILP is found we report the final genotype consisting of all alleles i such that $a_i > 0$ in the optimal solution. The copy number of each included allele i is set to a_i.

Although they are extremely unlikely, there are very few cases where discerning between different *CYP2D6* genotypes is theoretically infeasible using typical HTS data. In these cases, the proposed ILP has more than one optimal solution resulting in at least 2 different but equally likely genotypes. These cases occur when there are two sets of alleles, denoted A_1 and A_2, satisfying the following conditions:

1. The union of variations defining alleles from A_1 is identical to the union of variations defining alleles from A_2.

2. Ambiguous variations from those sets are not close enough to be covered by the paired-end reads originating from only one allele.

One simple example for this is shown in Figure 6.4. In this figure, the presence of depicted SNPs can signal the genotype combination of either *34/*6C or *2D/*6B. Since key SNPs for discerning these two possibilities are more than 1000bp apart, we cannot resolve this ambiguity using typical paired-end read data.

Note that we can enumerate all possible optimal solutions by using the API interface provided by the solver. If this interface is not available, we can obtain alternate optimal solutions by adding the additional set of constraints that makes the current solution infeasible. For example, if the optimal solution is $a_1 = 2$, we can find additional optimal solutions by solving two instances of the abovementioned problem: one with the additional constraint $a_1 < 1$, and one with the constraint $a_1 > 1$. As long as the value of the objective function remains the same, this process can be recursively applied for any additional optimal solution.

In such cases we report *all* of the most likely genotypes together with a *warning* that there is ambiguity in the inferred genotype. If these genotypes result in different phenotypes, further sample analysis is required. With the help of upcoming advances in HTS technologies and the increase in read lengths and insert sizes, we expect to resolve this problem in the near future.

6.2 Results

The first set of validated *CYP2D6* genotypes have recently been made available [45] for publicly available HTS data from 1000 Genomes Project Phase I collection [1]. Unfortunately, none of these samples are suitable for our purposes as they are either sequenced at very low coverage (2–5×) or have very short read length (36bp). Our method requires a minimum coverage of 10× per strand in order to successfully filter out the noise originating from sequencing errors. Furthermore reads longer than 60bp can ensure unambiguous mapping, as described in the filtering step of the Methods section. As a result, proper *CYP2D6* genotype data for publicly available HTS experiments with reasonable coverage and read length is not yet available.

In order to evaluate the performance of Cypiripi, we custom designed benchmark data consisting of the following:

1. **Simulation data:** Cypiripi was evaluated on 71 simulated datasets designed to reflect known *CYP2D6* genotypes [94], including theoretically possible but highly unlikely cases;

2. **Real data:** Cypiripi was evaluated on publicly available CEPH 1463 trio (mother, father and son) sequenced by Illumina HiSeq 2000 platform with average coverage of 100× per chromosome.

6.2.1 Simulations

Five sets of simulations, each covering a unique class of *CYP2D6* allelic arrangement, were created for evaluating the performance of Cypiripi. Those arrangements were constructed with the aim of covering all possible allelic combinations, including copy number changes and fusion events, as depicted in Figure 6.1. Within each set, we simulated several individuals with the set's specific allelic arrangement. The sets are defined as follows:

a) diploid case where both maternal (M) and paternal (P) chromosomes have the allele of the same type (e.g. *1/*1);

b) diploid case where both chromosomes contain one allele each and the alleles are different (e.g. *1/*3A);

c) both chromosomes contain a common tandem duplication or deletion event (e.g. 5×*2X/5×*2X or *5/*5); note that *5 allele describes a *CYP2D6* deletion;

d) both chromosomes contain a common variety of different alleles (e.g. *1E *14B *2X *14A for every chromosome);

e) both chromosomes contain a *CYP2D7* fusion or conservation event (e.g. *13A/*13A or *4A *68A/*4A *68A).

Note that Cypiripi reports the total number of alleles found in an individual genome without making distinction between chromosomes (e.g. *1/*1 will be reported as 2×*1).

Also note that set (d) is quite unrealistic, since such cases with large number of distinct variants are yet to be observed. We include these samples in order to show the generality of the method, and to evaluate its ability to cope with complex cases which could be encountered in the future. Sets (c) and (e), on the other hand, were specifically designed to reflect some of the previously discovered and validated genotypes [94].

For every sample, we separately constructed the sequences of maternal and paternal chromosomes, based on chromosome 22 of human reference genome, version hg38. We have inserted in each chromosome the corresponding *CYP2D6* gene within the coordinates of chr22:42,122,966–42,132,410. We have also replaced *CYP2D7* with some of the randomly selected *CYP2D7* genes mentioned in the Chapter 6.1, in order to account for *CYP2D7* variability between different individuals. In the case of fusions and duplications, we have followed the guide from [94], as depicted in Figure 6.1.

Simulated reads were generated by using simNGS (http://www.ebi.ac.uk/goldman-srv/simNGS/) simulator, which is capable of accurately simulating Illumina HiSeq 2000 machine parameters (details in the Appendix C), including substitution and indel rate. We have generated 101bp paired-end library with the average insert size of 400. Paired-end

coverage per chromosome was around 20×, totalling average coverage of 40× per individual. Although the current standard is approaching 200× per individual, we opted for lower coverage in order to show the robustness of the method.

Table 6.1: Cypiripi performance for the first three simulation groups. Correctly identified alleles are shown in green, while incorrect estimates are reported in red colour. In case of mismatches, red colour is also applied to the second column items in order to pinpoint the problematic allele. Unless otherwise specified, genotypes are given for both maternal and paternal chromosome in the format M/P. For ambiguous cases, all optimal genotypes are reported (e.g. 26th sample).

Set (a) Diploid cases with the same allele			**Set (b)** Diploid cases with different alleles		
$\lambda = 20, \eta = 8$			$\lambda = 20, \eta = 8$		
ID	Allele M/P	Result	ID	Allele M/P	Result
01	*1/*1	✓/✓	20	*6D/*55	✓/✓
02	*15/*15	✓/✓	21	*65/*53	✓/✓
03	*4M/*4M	✓/✓	22	*39/*73	✓/✓
04	*6A/*6A	✓/✓	23	*101/*45A	✓/✓
05	*27/*27	✓/✓	24	*2H/*1	✓/✓
06	*40/*40	✓/✓	25	*2B/*30	✓/✓
07	*10A/*10A	✓/✓	26	*6B/*2D	✓/✓or *6C/*34
08	*2K/*2K	✓/✓	27	*44/*2G	✓/✓
09	*2X/*2X	✓/✓	28	*71/*4M	✓/✓
10	*9/*9	✓/✓	29	*18/*62	✓/✓
11	*103/*103	✓/✓	30	*1C/*1B	✓/✓
12	*105/*105	✓/✓	31	*32/*25	✓/✓
13	*21B/*21B	✓/✓	32	*46h1/*105	✓/✓
14	*20/*20	✓/✓	33	*4C/*84	*4E/✓
15	*3B/*3B	✓/✓	34	*6C/*72	✓/✓
16	*28/*28	✓/✓	35	*28/*9	✓/✓
17	*1E/*1E	✓/✓	36	*3A/*8	✓/✓
18	*4G/*4G	✓/✓	37	*35X/*85	✓/✓
19	*38/*38	✓/✓	38	*2K/*3B	✓/✓

Set (c) Duplication and deletion events					
$\lambda = 20, \eta = 10$					
ID	Allele M/P	Result	ID	Allele M/P	Result
39	2×*35X/2×*35X	✓/✓	47	4×*2X/4×*2X	✓/✓
40	2×*4A/2×*4A	✓/✓	48	4×*1/4×*1	✓/✓
41	2×*9X/2×*9X	✓/✓	49	5×*2X/5×*2X	✓/✓
42	2×*10A/2×*10A	✓/✓	50	5×*1/5×*1	✓/✓
43	2×*2X/2×*2X	✓/✓	51	8×*2X/8×*2X	✓/7×*2X
44	2×*1/2×*1	✓/✓	52	8×*1/8×*1	✓/✓
45	3×*2X/3×*2X	✓/✓	53	8×*17/8×*17	✓/✓
46	3×*1/3×*1	✓/✓	54	*5/*5 (deletion)	✓/✓

All simulation results are shown in Tables 6.1 and 6.2. Cypiripi performed extremely well, providing 100% correct genotype for majority of the cases (62 out of 71). In 4 out of 9

Table 6.2: Cypiripi performance for the last two simulation groups. For set (d), where both chromosomes have the same allelic combination, we only show content of one chromosome for the sake of brevity. Results for set (d) are still reported for each chromosome separately. The rest of the table is organized as Table 6.1.

Set (d) Multiple copies of various types (both chromosomes reported once)		
$\lambda = 20, \eta = 8$		
ID	Allele M/P	Result
55	*4C *14A *75 *74 *37 *21B *20	✓/*4K,✓
56	*24 *4L *71 *103 *18 *70 *14A	✓/*4E,✓
57	*54 *46h2	✓/✓
58	*2D *65 *86 *43 *73 *25 *4K	✓/4K,✓and one *86 missing
59	*1E *14B *2X *14A *35A *45A *48	*14A,✓/*2X,✓
60	*37 *26 *4G	✓/✓
61	*103 *22 *2D	✓/✓
Set (e) Fusions and conservations with *CYP2D7*		
$\lambda = 20, \eta = 8$		
ID	Allele M/P	Result
62	*13A/*13A	✓/✓
63	*1 *13A/*1 *13A	✓/✓
64	*13C/*13C	✓/✓
65	*13D *2A/*13D *2A	✓/✓
66	*2A/*2A	✓/✓
67	2×*1 *13H/2×*1 *13H	✓/✓or 2D7/2D7
68	*36S/*36S	✓/✓
69	*82/*82	✓/✓
70	*4A *68A/*4A *68A	✓/✓
71	*10A *57/*10A *57	✓/*10D *57

remaining cases, Cypiripi reports an allele belonging to the same family as the correct allele (e.g. *4E and *4C from sample 33 belong to the same family *4). Copy number estimation was not in agreement with the ground truth in only two cases (samples 51 and 58), both having very large *CYP2D6* copy number (16 and 14, respectively). In these two cases the inferred copy number was lower by one compared to the ground truth. In all other cases, copy number was identified properly. Sample 26 contains ambiguous genotype described in Methods section, and in this case Cypiripi reported both genotypes as equally likely.

All samples from Tables 6.1 and 6.2 contained two copies of *CYP2D7* (one for maternal and for paternal chromosome), excluding the samples containing *13 allele (because all *13 fusions imply the removal of *CYP2D7*). The number of *CYP2D7* genes was estimated correctly in all samples.

Cypiripi has a special mode to detect and resolve fusion cases. The main difference consists of less stringent filtering used for samples containing fusions and conservations, since such alleles contain the same set of uncertain variations as *CYP2D7*. It is important to stress, as can be seen from the set (e) in the Table 6.2, that Cypiripi is able to successfully

handle various fusion cases. The only problematic case is misdetection of *13F and *13H as *CYP2D7*. Unfortunately, these fusions occur at the end of exon 9, preserving majority of *CYP2D7* and just a small portion of *CYP2D6*1*. Since all *CYP2D7*-specific variations are present in *13F and *13H, Cypiripi might detect either *CYP2D7* or *13F/H. Due to the fact that all *13 alleles encode the poor metabolizer as does *CYP2D7*, the corresponding phenotype is still accurately assigned based on the reported genotype.

We set the filtering threshold parameter η to be $0.4 \times \lambda$. Higher values perform better when the copy number is very high. Thus, we used $\eta = \lambda/2$ for the set (c).

It is worth mentioning that Cypiripi is a highly optimized and efficient tool. It requires only few minutes for a simple sample with two *CYP2D6* copies, and no more than 10 minutes on any other sample we evaluated on Intel Xeon 3.50 GHz CPU. This makes it ideal choice for clinical environments where the speed is of high importance.

6.2.2 Real data

In order to evaluate the performance of Cypiripi on real data sets, we used the family trio from CEPH 1463 pedigree. This trio consists of mother, father and son with high coverage Illumina HiSeq 2000 sequencing data publicly available for each of its members (http://www.illumina.com/platinumgenomes/). In addition to the sequencing data, the highly confident SNPs for NA12878 (mother), which belongs to this trio, were identified [194]. The analysis of these SNPs confirmed the presence of two *CYP2D6* copies. The first copy was validated as *CYP2D6*3A* and the obtained signal allows for the validation of second copy up to the allelic family level (*CYP2D6*4*). A genotype inferred by Cypiripi is in the agreement with both of these results. Namely, it was able to accurately identify the existence of *CYP2D6*3A* and reported the second copy as *CYP2D6*4M*.

Cypiripi reported *4M/*4M as a genotype for both father and son (Table 6.3). Although we don't have ground truth about *CYP2D6* genotypes for these two individuals, these predictions are in the strong agreement with Mendelian laws of inheritance.

Table 6.3: Cypiripi predictions for the real data set. NA12878 predictions are coloured in green due to the fact that they match the highly confident SNP calls from [194]. Since the validated predictions are not available for the other two samples, predictions of their genotypes are coloured black.

Real data samples CEPH 1463 trio	
ID	Identified
NA12877 (father)	***4M/*4M**
NA12878 (mother)	***3A/*4M**
NA12882 (son)	***4M/*4M**

The coverage parameter λ for those samples was set to 100, with the exception of NA12877, whose measured coverage was lower and was equalling 90. The η was, as it was the case with the simulated samples, set to $0.4 \times \lambda$.

6.3 Conclusion

In this chapter we have presented the first computational framework to exactly characterize the clinically important *CYP2D6* gene and its variations by using HTS data only. Our framework, which we call Cypiripi, is able to cope with many of the issues presented by the existing (non HTS based) approaches for *CYP2D6* genotyping, such as their inability to perform accurate copy number estimation, *CYP2D7* variant characterization and fusion detection.

In addition, Cypiripi's highly optimized running time makes it an ideal choice for clinical settings where speed is of high importance. The algorithmic basis of Cypiripi, a gene-agnostic integer linear program (ILP), can be easily extended to other unique gene clusters with similar properties.

It should be noted that there remain some challenges that we aim to investigate in follow-up work. For example, genotyping when the available set of sequence variants can be described by more than one set of genotypes is problematic. This technology-bound issue can be resolved by the use of paired-end reads in some cases but may require the availability of longer reads for the resolution of other cases. In addition, exact characterization of novel genotypes within the *CYP2D* locus is a further goal to be investigated.

As the cost of Whole Genome Sequencing (WGS) plummets and approaches the cost of exome sequencing, we will be able to perform detailed sequence analysis of several clinically important loci across the human genome by using standard coverage HTS data. This can reduce both the time and cost required for genomic analysis and address many of the limitations of existing (non-HTS based) techniques.

As whole genome sequencing makes its way into the clinic, it is providing economical and efficient means to identify many pharmacogenomical variants that can be used to provide personalised medication options. By the use of a proper computational framework such as Cypiripi, decision support systems to assist physicians for prescribing specific medications can benefit from fast and accurate genotyping based on HTS.

Chapter 7

Exact genotyping of ADMER genes using PGRNseq sequencing data

The advances in DNA sequencing over the past two decades made it possible to explore the human genome at unprecedented detail. The whole genome sequencing (WGS) is nowadays routinely performed in less than a day, and the recently introduced Illumina HiSeq X HTS sequencer pushed the cost of WGS under $1000 dollars per sample. Furthermore, Illumina-style WGS data offers high coverage depth, uniform read distribution and low error rates, all of which are useful for genotyping purposes. However, WGS is still considered costly and time-consuming compared to the commonly used targeted genotyping panels. Whole exome sequencing (WES) provides cheaper alternative to WGS, but in its current iteration it is not able to sequence non-coding regions. This makes WES unsuitable for genotyping of pharmacogenes, where variations in the non-coding regions can significantly affect phenotype [111].

Targeted genotyping platforms, like Affymetrix DMET+ arrays and the Illumina ADME assays, are able to detect the common set of predefined variations and genotypes. However, rare variations are common across the sites which impact the drug response [128]. Thus, Pharmacogenomics Research Network (PGRN), with the help of three large-scale sequencing centres (Department of Genome Sciences at University of Washington, The Genome Institute at Washington University, and the Human Genome Sequencing Center at Baylor College of Medicine), recently developed a sequencing panel named PGRNseq [59]. This panel targets 84 genes of pharmacogenomical interest (also known as ADMER genes), including genes encoding drug-metabolizing enzymes, drug transporters and drug targets. For each of these genes, PGRNseq sequences at least its exonic region and few kilobases upstream and downstream of gene's UTR region. In addition, PGRNseq keeps backwards compatibility with previous DMET+ and ADME assays by targeting all single nucleotide

variations (SNV) included in those panels. In total, more than 960 KB of genome is covered by PGRNseq. PGRNseq is itself based on Illumina HiSeq 2000 platform, and provides low error rates while maintaining very high depth of coverage (average of 500× per chromosome). Most importantly, PGRNseq is significantly cheaper than WES or WGS. For example, PGRNseq is up to ten times cheaper compared to WGS. Thus, PGRNseq offers a competitively priced platform for clinical genotyping of targeted genes while providing all benefits of standard WGS sequencing.

However, PGRNseq also inherits some of the problems that come with WGS, which include short read length and data interpretation issues. Genotype inference for ADMER genes harbouring various structural rearrangements, like *CYP2D6* and *CYP2A6*, still presents a major challenge. In order to assist the analysis of such structural variants, the second iteration of PGRNseq covers the whole genic clusters which contain those two genes (e.g. for *CYP2D6*, the whole 30 KB *CYP2D* cluster which includes *CYP2D6* and pseudogenes *CYP2D7* and *CYP2D8* is sequenced). Additional obstacle introduced by PGRNseq is non-uniformity of the coverage, which, despite its depth, further complicates detection of structural rearrangements.

These issues are especially relevant since *CYP2D6* is itself involved in metabolism of 20–25% of clinically prescribed drugs [81]. Structural variations of *CYP2D6*, which include the gene deletion, duplications and fusions with neighbouring *CYP2D7* can significantly affect the *CYP2D6* enzyme activity [136]. Cytochrome P450 2A6 (*CYP2A6*) also metabolizes several clinically used drugs, but more importantly, it is the principle metabolizer of nicotine and its by-product cotinine. It has been suggested that *CYP2A6*'s genotype is correlated with lower smoking risks, decreased cigarette consumption [161] and increased cessation [63]. Similar to *CYP2D6*, *CYP2A6* is highly polymorphic with more that 50 alleles observed so far [166], and it also harbours various gene duplications and crossovers with highly homologous neighbouring pseudogene *CYP2A7*. For these reasons, proper genotyping of *CYP2D6* and *CYP2A6* can predict the patient's response to some drugs, as well as shed some light on patient's smoking behaviour.

So far, no available computational tool is capable of inferring *CYP2D6* and *CYP2A6* genotypes from PGRNseq data. Previous *CYP2D6* genotyping tools, like Cypiripi [132] and Constellation [176], are either designed for uniform coverage WGS data, or are not able to properly detect some structural rearrangements (e.g. both Cypiripi and Constellation are not able to detect non-functional *68 allele in CEPH 1463 samples). Moreover, neither of these tools provides support for *CYP2A6* genotyping.

In this chapter we present Cypiripi^{++}, a significant improvement upon our previous tool Cypiripi. Cypiripi^{++} is able to properly genotype *CYP2D6* and *CYP2A6* on both PGRNseq and Illumina WGS data. Moreover, Cypiripi^{++}'s modular design allows easy inclusion of other ADMER genes, while still being able to deal with various gene duplications, fusions and deletions. By evaluation Cypiripi^{++} on large selection of real data WGS and PGRNseq

samples, we show that it can be used as a fast and accurate tool for *CYP2D6* and *CYP2A6* genotyping.

7.1 Methods

Cypiripi^{++} consists of the following steps:

Read alignment and mutation detection step: where HTS reads are aligned to the reference genome and SNVs present in target gene region are identified;

Copy number detection step: where the copy number of each region is determined and, if present, various structural variations are identified;

Protein identification step: where all major allelic configurations which affect aminoacid selection (and thus the final protein product) are established; and

Genotype refinement step: where the supporting set of non-functional SNVs is used to rank each allelic configuration found in the previous three steps.

Final genotype is obtained by choosing the set of allelic configurations with the best ranking score. In case of multiple configurations with the same score, all will be reported as equally likely genotypes.

In addition to HTS data in SAM/BAM file format, the input to Cypiripi^{++} also contains a database having the information about the gene to be genotyped. This information includes list of alleles, their functional and non-functional mutations, as well as the information about the possible structural variations. For *CYP2D6* and *CYP2A6* genes considered in this work, we constructed the corresponding databases by using the data from The Human Cytochrome P450 Allele Nomenclature Database (`http://www.cypalleles.ki.se/`) and cross-validating it with dbSNP [165].

7.1.1 Read Mapping

The common practice is to map reads to the reference genome by following the "best practices workflow" [177], usually involving BWA read aligner [106] and Genome Analysis Toolkit (GATK) [120]. We have used this workflow for all evaluated samples. However, Cypiripi^{++} is not limited to this mapping framework, and accepts any valid SAM/BAM file which contains the region of the targeted gene. GATK pipeline is recommended since it performs the local indel realignment [32], which improves the detection of various small indels.

7.1.2 Copy number estimation

As we have already discussed in Chapter 5, *CYP2D6* is prone to copy number variations including gene deletions, duplications as well as multiplications. High sequence homology between *CYP2D6* and related pseudogene *CYP2D7* also results in the formation of gene conversions producing hybrid genes where one part of the gene originating from *CYP2D6* fuses with the other one originating from *CYP2D7*. In many cases the existence of these structural variations has a substantial impact on the resulting protein products and therefore their accurate characterization is highly important for the proper phenotype predictions. For the illustration purposes, in this section we will focus only on *CYP2D6*. Nevertheless, all models and calculations described below apply to other genes as well, including *CYP2A6*.

In Chapter 6, we showed how the above problem can be solved by using reads obtained from the sequencing technology which provides the data of uniform coverage (e.g. Illumina HiSeq), assuming that the depth of coverage is provided as user-defined parameter. However, this approach is not suitable for data generated by PGRNseq platform where coverage is highly non-uniform across regions and its depth is usually not known in advance.

By analyzing 96 samples sequenced by PGRNseq platform, we have observed that the depth of coverage usually follows the same shape across different samples as illustrated in Figure 7.1. In order to characterize this shape for PGRNseq data of an arbitrary sample S, we first consider the PGRNseq data for NA19686 individual. It is known that *CYP2D6* genotype of this individual consists of two reference *1 alleles. We introduce the function

$$B_g : \{1, 2, \ldots, |g|\} \rightarrow \mathbf{R},$$

where g and $|g|$ denote the gene of interest (in our case *CYP2D6*, *CYP2D7* or *CYP2D8*) and its length, respectively. The value of $B_g(i)$ equals to the sum of coverage depths of both chromosomal copies for i-th nucleotide of gene g in NA19686.

Sequencing experiments generating data for S and NA19686 are not necessarily equivalent in terms of depth of coverage. Consequently, we need an appropriate rescaling of function B_g in order to obtain the function of reference coverage depth for the sequencing experiment of sample S. Analogously to B_g, we define this function as

$$R_g : \{1, 2, \ldots, |g|\} \rightarrow \mathbf{R}.$$

Intuitively, $R_g(i)$ represents a depth of coverage for i-th nucleotide of NA19686 sequenced under the same conditions as the sample S. As B_g and R_g follow the same shape we can estimate R_g as

$$R_g(i) = \eta \times B_g(i), \qquad \forall i \in \{1, 2, \ldots, |g|\}$$

where η is the ratio of sequencing depths of the two experiments.

In order to estimate η, we use B_g and depth of observed coverage function for sample S, here denoted as C_g and defined analogously to R_g and B_g. Using region q of stable copy number that is not involved in any structural variations we can estimate η as

$$\eta = \frac{C_g(q)}{B_g(q)},$$

where $C_g(q)$ and $R_g(q)$ are obtained by summing all values of $C_g(i)$ and $R_g(i)$, respectively, for any i falling into the region q.

One of the regions from *CYP2D* locus having this property is the region of *CYP2D8* containing exons 4, 5 and 6. Using this region as q in the above formula leads to proper estimate of η that is later used for computing the reference coverage depth function R_g for sample S. Note that above R_g and C_g are not necessarily identical due to the possible presence of structural variations in sample S. Example of rescaling is given in Figure 7.1.

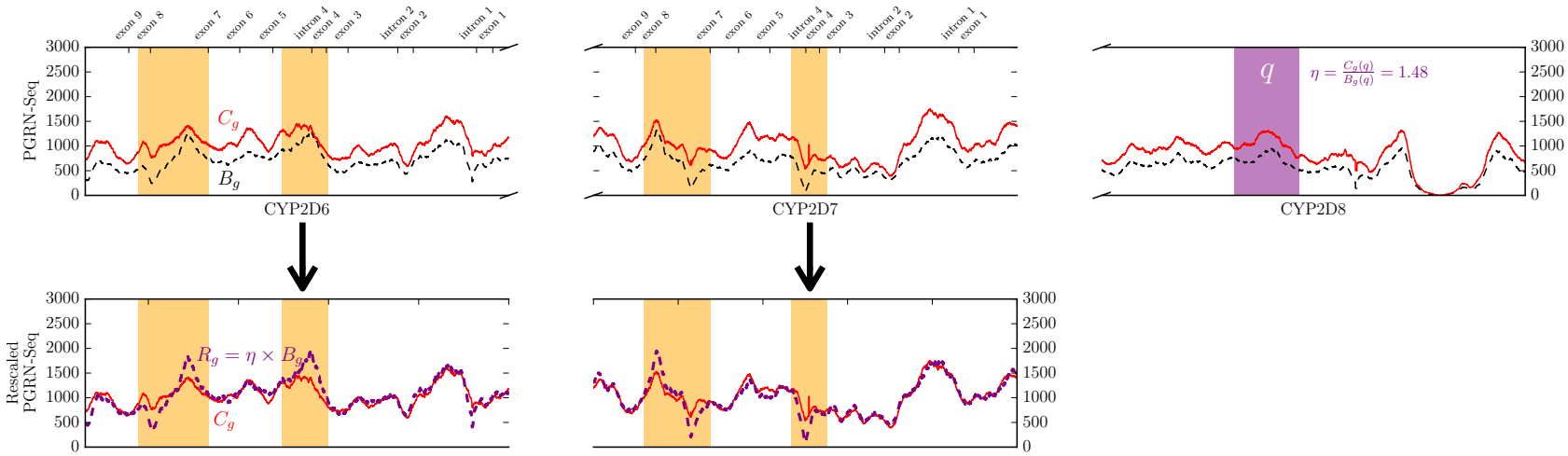
Due to the existence of rearrangement events, we cannot directly work on estimating copy number of regions as large as the whole genes. Namely, in the case of hybrid gene it is impossible to define the exact number of *CYP2D6* and *CYP2D7* whole-gene copies as only a portion of each gene is involved in the corresponding fusion event. This motivates us to consider a copy number status of smaller regions that would allow proper characterization and detection of these complicated cases. For this purpose, let g and h stand for *CYP2D6* and *CYP2D7*, respectively, and assume that we split g and h into regions $r_1, r_2, \ldots, r_n$ and $r'_1, r'_2, \ldots, r'_n$, respectively. In order to characterize rearrangement configurations, we introduce a binary vector $\mathbf{v}$ consisting of $2n$ entries which is defined as follows:

$$\mathbf{v}[i] = \begin{cases} 0, & \text{if } i \leq n \text{ and } r_i \text{ is not present in a given configuration} \\ 0, & \text{if } i > n \text{ and } r'_i \text{ is not present in a given configuration} \\ 1, & \text{otherwise.} \end{cases}, \tag{7.1}$$

Considering *CYP2D* locus at the single chromosome, note that the most frequent case where one copy of each of *CYP2D6* and *CYP2D7* is present can be simply represented as a vector $\mathbf{v}$ having all entries equal to 1. Deletion case can be represented as a vector $\mathbf{v}$ consisting of n zeroes followed by n ones, whereas multiplication case containing k copies of *CYP2D6* without hybrid genes can be represented as a sum of $[1, 1, \ldots, 1]$ and $k-1$ vectors $[1, 1, \ldots, 1, 0, 0, \ldots, 0]$, where the first n entries of the last vector are equal to 1 and the remaining n entries are equal to 0. Some examples considering *CYP2D* locus on both autosomes and covering some more complicated cases including hybrid genes are shown in Figure 7.2.

Let $M = \{\mathbf{v}_m\}$ be a set of possible configurations constructed from *CYP2D6* online database. For each region $r = [a, b]$, denote by $\mathbf{c}[r]$ its observed coverage in a normalized

Figure 7.1: Example of PGRNseq coverage rescaling for some sample S. Red line indicates the coverage of sample S, C_g, while dashed black line indicates NA19686 coverage B_g. Purple dashed line indicates rescaled $R_g = \eta \times B_g$. Purple shade denotes the region q from *CYP2D8*. Identical regions are regions are shaded in orange colour.



form calculated as follows:

$$\mathbf{c}[r] = 2 \times \frac{\sum_{a \leq i \leq b} C_g(i)}{\sum_{a \leq i \leq b} R_g(i)}.$$

Factor 2 is included to account for both autosomes present in the real data sets.

Our goal is to find a set $M_{opt} \subseteq M$ such that the difference between the observed coverage and the coverage formed by M_{opt} is minimal. We can model this problem as binary integer programming as follows: let v_m be a binary variable which is 1 if and only if $\mathbf{v}_m \in M$ is included in M_{opt}. We aim to minimize

$$\min \sum_r |E_r| + \lambda, \tag{7.2}$$

subject to:

$$E_r = \mathbf{c}[r] - \sum_{\mathbf{v}_m \in M} \mathbf{v}_m[r] \times v_m \quad \text{for each region } r \in \{r_1, \ldots, r_n, r'_1, \ldots, r'_n\}.$$

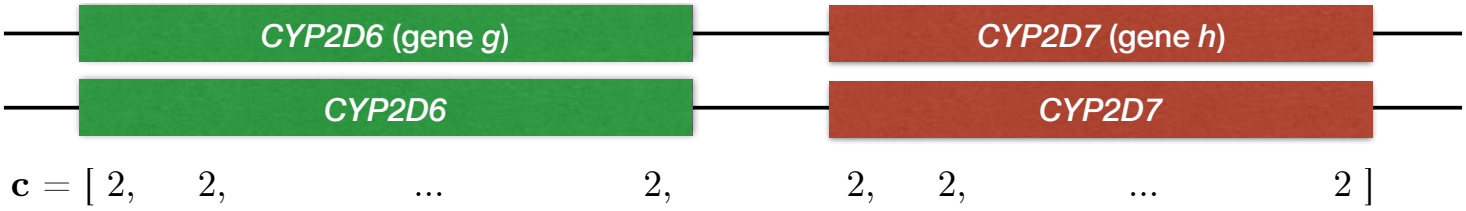
In Equation (7.2), λ indicates the control variable. We set $\lambda = 0.1 \sum_{m \in F} v_m$, where F represents a set of indices i such that $\mathbf{v}_i$ is hybrid gene configuration. We introduce this term in order to avoid various ambiguities caused by fusion rearrangements.

Although we use binary variables in the Equation (7.2), multiple copies of one allele can be easily modelled as a set of multiple binary variables. The same reasoning is used in the models described below for the sake of explanation.

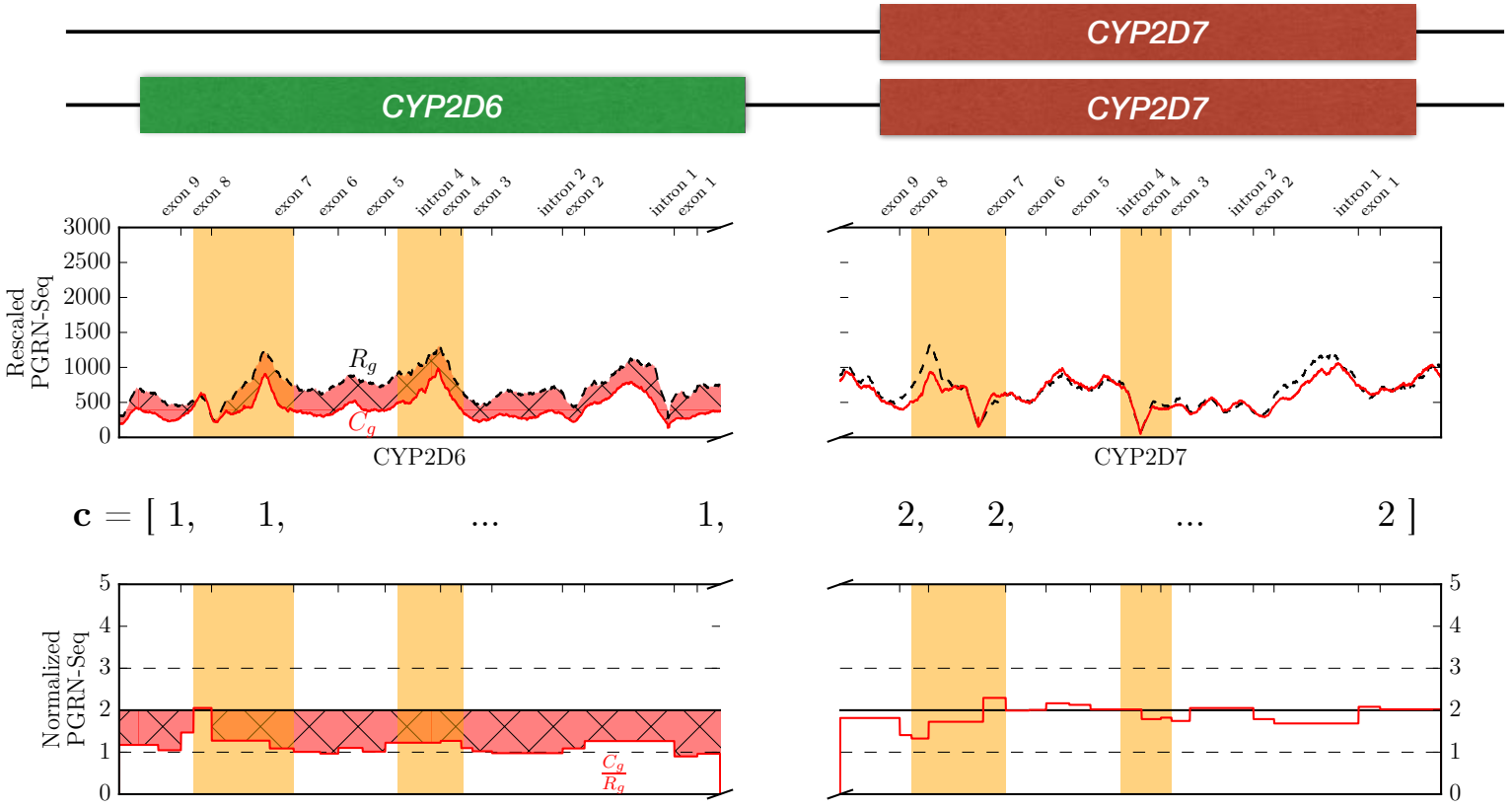
Various sequencing errors and misalignments can introduce some variance during the calculation of coverage vector $\mathbf{c}$ across the samples. We have noticed that the effect of such variance does not impact the abovementioned model as long as the depth of coverage is greater than 20×. However, this does not hold for few regions in *CYP2D6* and *CYP2D7* which are identical (e.g. intron 7 or exon 8). Cypiripi^{++} does not include these regions in the analysis performed above, because many misaligned reads originating from these regions significantly affect the accuracy of $\mathbf{c}[r]$. Impact of the misaligned reads is clearly visible in the Figure 7.2, where the identical regions are shaded with orange colour. There are only a few alleles having breakpoints in these regions. However, in these cases Cypiripi^{++} will still identify the presence of fusion events and predict the correct phenotype. For example, allele *CYP2D6*13G1* with the breakpoint in intron 7 will be detected as *CYP2D6*13E* that is having a breakpoint in exon 5. Both of these alleles represent a non-functional fusion with *CYP2D7*.

There might be multiple rearrangements M_{opt} which minimize Equation (7.2). Cypiripi^{++} will try to detect a genotype for each such optimal rearrangement, and pick the rearrangement whose genotype is most likely based on the subsequent steps.

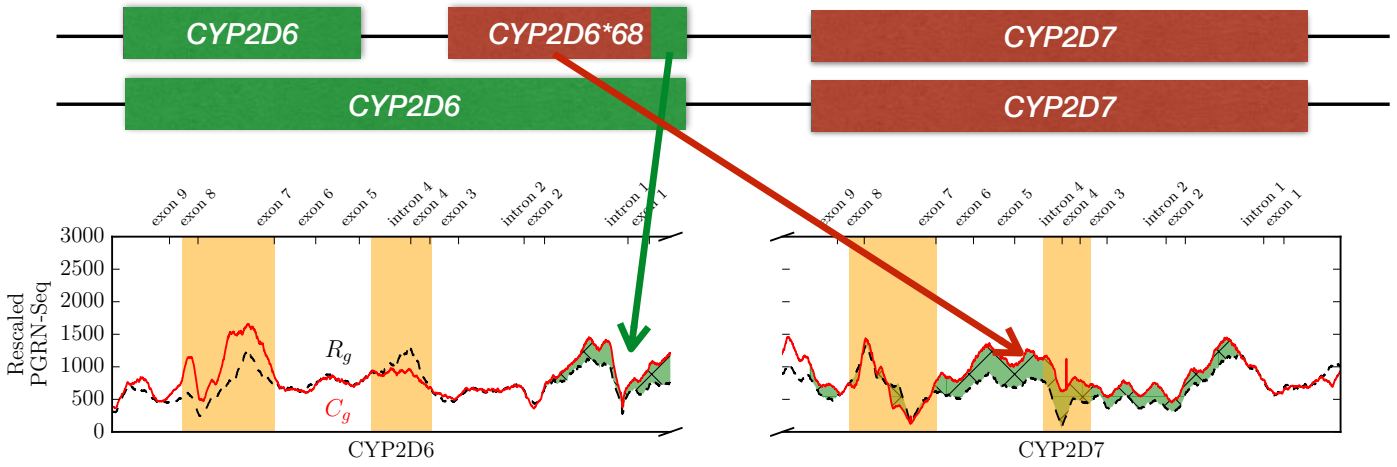
Figure 7.2: PGRNseq coverage normalization for three *CYP2D6* gene arrangements. For each case, first row indicates the rescaled PGRNseq coverage, while the second row indicates normalized coverage (i.e. C_g/R_g). Regions coloured with red denote deletion of *CYP2D6*, while green-coloured regions indicate duplication. Identical regions are shaded in orange colour. (i) Normal arrangement consisting one copy of each of *CYP2D6* and *CYP2D7* on both chromosomes. (ii) *CYP2D6* deletion on one chromosomal copy; (iii) one copy of *CYP2D6*1* accompanied by *CYP2D7/2D6* fusion (*68 allele) with the breakpoint in intron 1 on one chromosomal copy. Note the changes in vector **c** which describes each copy number structure. In the last example, set of vectors **v** which most closely describe vector **c** is given under the figure.



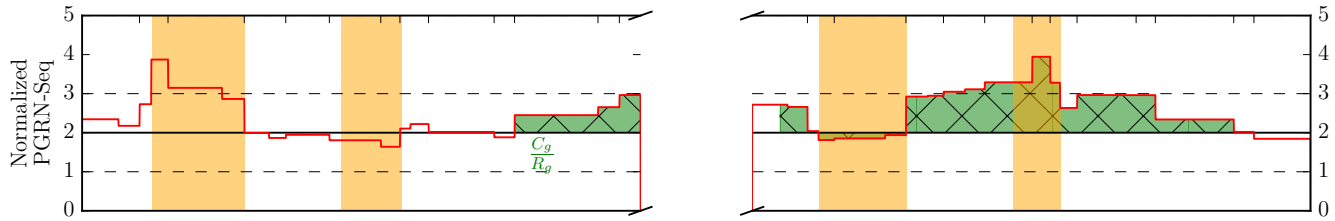
(i) Regular case


CYP2D7
CYP2D6
CYP2D7
exon 9
exon 8
exon 7
exon 6
exon 5
intron 4
exon 4
exon 3
intron 2
exon 2
intron 1
exon 1
Rescaled PGRN-Seq
3000
2500
2000
1500
1000
500
0
R_g
C_g
CYP2D6
CYP2D7
$\mathbf{c}$ = [1, 1, ... 1, 2, 2, ... 2]
Normalized PGRN-Seq
5
4
3
2
1
0
$\frac{C_g}{R_g}$

(ii) Deletion



$\mathbf{c} = [\ 2,\ \ 2,\ \ \ldots\ \ 2\ ,\ 3,\ \ 3,\ \ \ 3,\ \ 3,\ \ \ldots\ \ 3,\ \ 2,\ \ 2\ ] =$



$(\mathbf{v_1})$ [1, 1, ... 1 , 1, 1, 1, 1, ... 1, 1, 1] (regular) +
$(\mathbf{v_2})$ [1, 1, ... 1 , 1, 1, 1, 1, ... 1, 1, 1] (regular) +
$(\mathbf{v_3})$ [0, 0, ... 0 , 1, 1, 1, 1, ... 1, 0, 0] (*68 fusion in intron 1)

(iii) Fusion and duplication

7.1.3 Protein identification

One of the major goals of genotyping is an accurate phenotype prediction. For non-reference alleles (i.e. all alleles except *CYP2D6*1*), this problem can be reduced to detecting all of the variations that alter the final protein product. These include codon-changing SNPs, indels within the coding regions, some SNPs in the splicing regions and many others. We name any such variation *functional mutation*, and all other variations *non-functional mutations*. The set of functional mutations is already established in the *CYP2D6* allele database.

Cypiripi^{++} will first identify all protein products which can be produced by alleles in the database. Proteins which are not in agreement with the copy number configuration obtained from the previous step will get excluded. Each remaining protein can be represented as a set P_i of the associated functional mutations m_j.

Let us assign every remaining protein product P_i a binary variable p_i. This variable is set to 1 if and only if protein P_i is encoded by some of the alleles present in the genotype of our sample. For each functional mutation m_j present at the loci j, we can calculate

$$E_{m_j} = 2 \times \frac{C_g(m_j)}{R_g(j)} - \sum_{i:m_j \in P_i} p_i,$$

where $C_g(m_j)$ denotes the observed coverage of mutation m_j.

We need to enforce that each expressed functional mutation is included in some protein. This can be achieved by adding the following constraint:

$$\sum_{i:m_j \in P_i} p_i \geq 1 \qquad \text{for each expressed functional mutation } m_j.$$

Our objective is to select a set of proteins which most closely match the observed set of functional mutations. More formally, we aim to minimize:

$$\min \sum_{m_j} \left| E_{m_j} \right|. \tag{7.3}$$

In order to avoid the presence of alleles which contain non-expressed functional mutations, we require that

$$\sum_{i:m_j \in P_i} p_i \leq 0 \qquad \text{if } C_g(m_j) = 0.$$

This model can also produce multiple optimal solutions. Due to the short length of the PGRNseq reads (100bp) and short insert size (around 300bp), Cypiripi^{++} is not able to precisely resolve the cases where one set of distant mutations (i.e. mutations that cannot be spanned by read pairs) describes multiple protein products. Each such optimal protein product will be passed to the refiner step, where the final verdict will be made.

7.1.4 Genotype refining

From the biological standpoint, all proteins identified in the previous step can represent a correct (possibly not yet observed) solution. However, we found that proteins which do not form the correct solution can be eliminated by using non-functional variants from the allele database as a supporting evidence. We will call the inclusion of non-functional variations in the genotyping process as *genotype refining*.

The refining model is similar to the one used in the protein identification step. Let A_i denote the set of all mutations describing the i-th allele in the database. For each A_i, let us introduce the binary variable a_i controlling its presence in the final solution.

We also introduce a binary variable $e_{i,j}$ for every allele A_i and for every $m_j \in A_i$. This variable is set if and only if m_j is assigned to A_i during the optimization step. We also include a binary variable $f_{i,j}$ for any $m_j \notin A_i$, whose value is set to 1 if and only if allele A_i harbours mutation m_j in the optimal genotype. The role of variables e and f is to model the variability of non-functional mutations by allowing any allele to either lack some mutation specified in its definition, or to include some non-functional mutation not present in the database. This is grounded in the observation that many alleles examined in the real data samples are having some additional non-functional mutations not present in the database description of the allele. Clearly, both $e_{i,j}$ and $f_{i,j}$ are set to 1 only if a_i is also set to 1.

In this model, we again try to minimize the difference of observed and expected coverage. In a similar manner to the protein identification step, this difference can be expressed as

$$E_{m_j} = 2 \times \frac{C_g(m_j)}{R_g(j)} \times \left(\sum_{i:\ m_j \in A_i} a_i e_{i,j} + \sum_{i:\ m_j \notin A_i} a_i f_{i,j} \right) \qquad \text{for all mutations } m_j.$$

We would also like to minimize the number of both missing and additional non-functional variations, in order to match the database data as closely as possible. Thus, the objective function is written as:

$$\min \sum_{m_j} \left| E_{m_j} \right| + \sum_i a_i \left[\alpha \sum_{j:\ m_j \in A_i} (1 - e_{i,j}) + \beta \sum_{j:\ m_j \notin A_i} f_{i,j} \right], \tag{7.4}$$

where α and β denote the penalty scores for missing and additional mutations. In our experiments, we used $\alpha = 4$ and $\beta = 1$, because it is more likely for allele to include some additional mutation than to lack an observed one.

We enforce that each allele is assigned all of its functional mutations as follows. Let F_i be the set of functional mutations for allele A_i. Clearly, $F_i \subseteq A_i$, and this requirement can

be expressed by the following constraint:

$$a_i \times \left(|F_i| - \sum_{j:\ m_j \in F_i} e_{i,j} \right) = 0 \qquad \text{for each allele } A_i,$$

where $|F_i|$ denotes the cardinality of set F_i.

We ascertain that no variation is over-expressed (i.e. allele must follow the copy number prediction at each step), and that no functional mutation can be included by $f_{i,j}$ (thus modifying the allele's protein product) with the additional sentinel constraints.

Finally, the set of protein products for which the Equation 7.4 gives the lowest score is selected as the final genotype. If multiple optimal solutions are found by the refiner, Cypiripi^{++} will report all of them as equally likely. One such example is *68+*4/*5, where both *68+*4/*5 and *68/*4 will have the same score. However, in this case phenotype prediction is not affected by such ambiguity.

7.1.5 Complexity

All of the models mentioned above can be expressed as the instances of the following, more general problem:

Problem 1. *Given a multiset of m-dimensional vectors $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, and a target vector $\mathbf{y} = [y_1, \ldots, y_m]$, find a minimal multi-subset $X' \subseteq X$ such that*

$$\sum_{i=1}^{n} \left| \mathbf{y}[i] - \sum_{\mathbf{x} \in X'} \mathbf{x}[i] \right|,$$

is minimized.

We can form a decision version of the Problem 1 by asking for a multi-subset X' such that

$$\mathbf{y}[i] = \sum_{\mathbf{x} \in X'} \mathbf{x}[i] \qquad \text{for every } i, \tag{7.5}$$

if such subset exists.

Theorem 2. *Problem 1 is NP-hard.*

Proof. As long as the decision version of Problem 1 is NP-complete, Problem 1 will be NP-hard. So we will prove that the decision version of Problem 1 is NP-complete by reducing the subset sum problem to it. Subset set problem is known to be NP-complete [86].

Consider a set $S = \{s_1, \ldots, s_n\}$ of integers, and sum s, which both describe the subset sum problem. Subset sum is trivially expressed as the instance of the Problem 1 by

setting $\mathbf{x}_i = [s_i]$ and $\mathbf{y} = [s]$ in the Equation (7.5). Clearly, any $X' \subseteq X$ which satisfies Equation (7.5) also implies

$$s = \mathbf{y}[1] = \sum_{\mathbf{x} \in X'} \mathbf{x}[1] = \sum_{i:s_i \in S'} s_i,$$

for some $S' \subseteq S$, and vice versa. Thus, the decision version of Problem 1 is NP-complete. □

Subset sum problem has a pseudo-polynomial dynamic programming solution of the complexity $O(ns)$. Similar dynamic programming can be devised for the decision version of Problem 1. In any instance of our interest, $\mathbf{x}_j$ consist only of integers, and $\mathbf{y}[i] \geq 0$ for any i. The adapted pseudo-polynomial dynamic programming solution will have the worst-case complexity of $O(nT^m)$, where T represents the maximum possible value of $y[i]$ for any i. However, note that instances described in the previous three sections are specialized versions of Problem 1 because of additional constraints introduced in their models. These constraints can significantly reduce the search space, and make the abovementioned problems solvable in practice.

7.2 Results

7.2.1 Experimental data

We have evaluated Cypiripi^{++} on 96 Coriell samples spanning 32 different family trios and multiple ethnic backgrounds. All samples were provided by Baylor College of Medicine. They were all sequenced on PGRNseq v2 platform with the average coverage of 600×. Genotypes of the sequenced samples contain many different *CYP2D6* and *CYP2A6* alleles, including those with various types of structural variations. Genotypes of all samples were validated with PCR-based genotyping panels. Cypiripi^{++}'s performance on those samples is shown in Table 7.1.

Table 7.1: *CYP2D6* genotypes inferred by Cypiripi^{++} on the set of 96 PGRNseq samples. Please refer to the discussion section for detailed explanation of the footnotes. F stands for father, M for mother, and C for child.

Sample ID	Family	Ethnicity	Gender	*CYP2D6* genotype	Cypiripi^{++} prediction
HG00421	SH007	Chinese	F	*2/*10×N	✓ (*2/*10+*10)
HG00422	SH007	Chinese	M	*2/*10	✓ (*2/*10)
HG00423	SH007	Chinese	C	*10/*10×N	✓ (*10/*10+*10)
HG00463	SH021	Chinese	F	*36+*10/*36+*10	✓ (*36+*10/*36+*10)
HG00464	SH021	Chinese	M	*1/*36+*10	✓ (*1/*36+*10)
HG00465	SH021	Chinese	C	*36+*10/*36+*10	✓ (*36+*10/*36+*10)
HG00592	SH057	Chinese	F	*1/*10	✓ (*1/*10)
HG00593	SH057	Chinese	M	*2/*36+*10	✓ (*2/*36+*10)

HG00594	SH057	Chinese	C	*1/*2	✓ (*1/*2)
HG01060	PR14	Puerto Rican	F	*1/*41	✓ (*1/*41)
HG01061	PR14	Puerto Rican	M	*1/*4	✓ (*1/*4)
HG01062	PR14	Puerto Rican	C	*1/*4	✓ (*1/*4)
HG01190	PR40	Puerto Rican	F	*68+*4/*5	✓ (*68+*4/*5)
HG01191	PR40	Puerto Rican	M	*2/*41	✓ (*2/*41)
HG01192	PR40	Puerto Rican	C	*5/*41	✓ (*5/*41)
HG01979	PEL027	Peruvian	F	*2/*68+*4	✓ (*2/*68+*4)
HG01980	PEL027	Peruvian	M	*1/*2	✓ (*1/*2)
HG01981	PEL027	Peruvian	C	*1/*2	✓ (*1/*2)
HG02259	PEL042	Peruvian	F	*1/*2	✓ (*1/*2)
HG02260	PEL042	Peruvian	M	*1/*1	✓ (*1/*1)
HG02261	PEL042	Peruvian	C	*1/*2	✓ (*1/*2)
NA06984	1328	European	F	*68+*4/*4	✓ (*68+*4/*4)
NA06989	1328	European	M	*9/*9	✓ (*9/*9)
NA12331	1328	European	C	*4/*9	✓ (*4/*9)
NA07357	1345	European	F	*1/*6	✓ (*1/*6)
NA07345	1345	European	M	*1/*1	✓ (*1/*1)
NA07348	1345	European	C	*1/*6	✓ (*1/*6)
NA10853	1349	European	F	*2/*41	✓ (*2/*41)
NA10854	1349	European	M	*1/*4	✓ (*1/*4)
NA11834	1349	European	C	*2/*4	✓ (*2/*4)
NA10860	1362	European	F	*1/*4	✓ (*1/*4+*4N)
NA10861	1362	European	M	*4/*2	✓ (*4/*35) [case (1)]
NA11984	1362	European	C	*1/*2	✓ (*1/*35) [case (1)]
NA11891	1377	European	F	*1/*1	✓ (*1/*1)
NA11892	1377	European	M	*6/*41	✓ (*6/*41)
NA10865	1377	European	C	*1/*41	✓ (*1/*41)
NA12003	1420	European	F	*4/*2 or *4/*35	✓ (*4/*35) [case (1)]
NA12004	1420	European	M	*2/*41	✓ (*2/*41)
NA10838	1420	European	C	*2/*4	✓ (*2/*4)
NA12155	1408	European	F	*1/*5	✓ (*1/*5)
NA12156	1408	European	M	*1/*4	✓ (*1/*4)
NA10831	1408	European	C	*4/*5	✓ (*4/*5)
NA12272	1418	European	F	*1/*1	✓ (*1/*1)
NA12273	1418	European	M	*1/*1	✓ (*1/*1)
NA10837	1418	European	C	*1/*1	✓ (*1/*1)
NA12342	1330	European	F	*4/*41	✓ (*4/*41)
NA12343	1330	European	M	*1/*5	✓ (*1/*5)
NA12336	1330	European	C	*5/*41	✓ (*5/*41)
NA12399	1354	European	F	*1/*1	✓ (*1/*1)
NA12400	1354	European	M	*1/*68+*4	✓ (*1/*68+*4)
NA12386	1354	European	C	*1/*1	✓ (*1/*1)
NA12750	1444	European	F	*2/*2	✓ (*2/*2)

NA12751	1444	European	M	*1/*2	✓ (*1/*2)
NA12740	1444	European	C	*1/*2	✓ (*1/*2)
NA12801	1454	European	F	*4/*6	✓ (*4/*6)
NA12802	1454	European	M	*2/*41	✓ (*2/*41)
NA12805	1454	European	C	*2/*4	✓ (*2/*4)
NA12891	1463	European	F	*68+*4/*41	✓ (*68+*4/*41)
NA12892	1463	European	M	*2/*3	✓ (*2/*3)
NA12878	1463	European	C	*3/68+*4	✓ (*3/*68+*4)
NA18507	Y009	Yoruban	F	*2/*4×N	✓ (*2/*4+*4)
NA18508	Y009	Yoruban	M	*2/*5	✓ (*2/*5)
NA18506	Y009	Yoruban	C	*2/*5	✓ (*2/*5)
NA18516	Y013	Yoruban	F	*1/*17	✓ (*1/*17)
NA18517	Y013	Yoruban	M	*5/*10	✓ (*5/*10)
NA18515	Y013	Yoruban	C	*1/*10	✓ (*1/*10)
NA19128	Y077	Yoruban	F	*17/*17	✓ (*17/*17)
NA19127	Y077	Yoruban	M	*2/*17	✓ (*2/*17)
NA19129	Y077	Yoruban	C	*17/*17	✓ (*17/*17)
NA19200	Y045	Yoruban	F	(*76)+*1/*5 or *1/*5	✓ (*1/*5) [case (2)]
NA19201	Y045	Yoruban	M	*1/*17	✓ (*1/*17)
NA19202	Y045	Yoruban	C	(*76)+*1/*1	✓ (*1/*1) [case (2)]
NA19239	Y117	Yoruban	F	*13-like?/*17 or *15/*17	✓ (*15/*17) [case (3)]
NA19238	Y117	Yoruban	M	*1/*17	✓ (*1/*17)
NA19240	Y117	Yoruban	C	*13-like?/*17	✓ (*15/*17) [case (3)]
NA19685	M011	Mexican-Am	F	*1/*2×2	✓ (*1/*2+*2)
NA19684	M011	Mexican-Am	M	*1/*4	✓ (*1/*4)
NA19686	M011	Mexican-Am	C	*1/*1	✓ (*1/*1)
NA19771	M031	Mexican-Am	F	*2/*4	✓ (*2/*4)
NA19770	M031	Mexican-Am	M	*1/*2	✓ (*1/*2)
NA19772	M031	Mexican-Am	C	*2/*4	✓ (*2/*4)
NA19789	M037	Mexican Am	F	*1/*1	✓ (*1/*1)
NA19788	M037	Mexican Am	M	*2/*78+*2	✓ (*2/*78+*2)
NA19790	M037	Mexican Am	C	*1/*78+*2	✓ (*2/*78+*2)
NA19700	2367	AA	F	*4/*29	✓ (*4/*29)
NA19701	2367	AA	M	*1/*17	✓ (*1/*17)
NA19702	2367	AA	C	*4/*17	✓ (*4/*17)
NA19818	2418	AA	F	*1/*17	✓ (*1/*17)
NA19819	2418	AA	M	*2/*4×2	✓ (*2/*4+*4)
NA19828	2418	AA	C	*2/*17	✓ (*2/*17)
NA19834	2424	AA	F	*2/*2	✓ (*2/*45) [case (4)]
NA19835	2424	AA	M	*1/*2	✓ (*1/*45) [case (4)]
NA19836	2424	AA	C	*1/*2	✓ (*1/*45) [case (4)]
NA19900	2425	AA	F	*3/*29	✓ (*3/*29)
NA19901	2425	AA	M	*1/*1	✓ (*1/*1)

NA19902	2425	AA	C	*1/*29	✓ (*1/*29)

In addition to the PGRNseq samples, we have also evaluated Cypiripi^{++} on the samples from the Platinum Genome project, which covers 17 individuals from CEPH 1463 family. All those samples were sequenced with Illumina HiSeq 2000 WGS sequencer with the average coverage of 50×. Furthermore, we included a few Illumina WGS samples from 1000 Genome project with the coverage exceeding 20× and available validations. Predictions made by Cypiripi^{++} on Illumina samples are shown in Table 7.2.

Table 7.2: *CYP2D6* genotypes inferred by Cypiripi^{++} on the set of 21 Illumina WGS samples. GF stands for grandfather, GM stands for grandmother, F stands for father, M for mother, and C for child. Samples for which the validation is missing are marked with N/A.

Sample ID	**Family**	**Ethnicity**	**Gender**	***CYP2D6* genotype**	**Cypiripi^{++} prediction**
NA19239	Y117	Yoruban	F	*15/*17	✓ (*15/*17) case (3)
NA19238	Y117	Yoruban	M	*1/*17	✓ (*1/*17)
NA19240	Y117	Yoruban	C	*15/*17	✓ (*15/*17) case (3)
NA19900	2425	AA	F	*3/*29	✓ (*3/*29)
NA12889	1463	European	GF	*4/*41	✓ (*4/*41)
NA12890	1463	European	GM	N/A	*68+*4/*68+*4
NA12891	1463	European	GF	*41/*68+*4	✓ (*41/*68+*4)
NA12892	1463	European	GM	*2/*3	✓ (*2/*3)
NA12877	1463	European	F	*4/*68+*4	✓ (*4/*68+*4)
NA12878	1463	European	M	*3/*68+*4	✓ (*3/*68+*4)
NA12879	1463	European	C	N/A	*3/*68+*4
NA12880	1463	European	C	N/A	*68+*4/*68+*4
NA12881	1463	European	C	N/A	*68+*4/*68+*4
NA12882	1463	European	C	*4/*68+*4	✓ (*4/*68+*4)
NA12883	1463	European	C	N/A	*3/*68+*4
NA12884	1463	European	C	N/A	*4/*68+*4
NA12885	1463	European	C	N/A	*68+*4/*68+*4
NA12886	1463	European	C	N/A	*3/*4
NA12887	1463	European	C	N/A	*4/*68+*4
NA12888	1463	European	C	N/A	*4/*68+*4
NA12893	1463	European	C	N/A	*3/*4

7.2.2 Discussion

As it can be seen from Table 7.1, Cypiripi^{++}'s predictions match the validated genotypes in most of the cases. Although we found several discrepancies between the predictions, after further investigation we concluded that genotyping panels made either ambiguous or incorrect calls. These cases are discussed below.

Table 7.3: *CYP2A6* genotypes inferred by Cypiripi^{++} on the set of 11 PGRNseq samples. F stands for father, M for mother, and C for child.

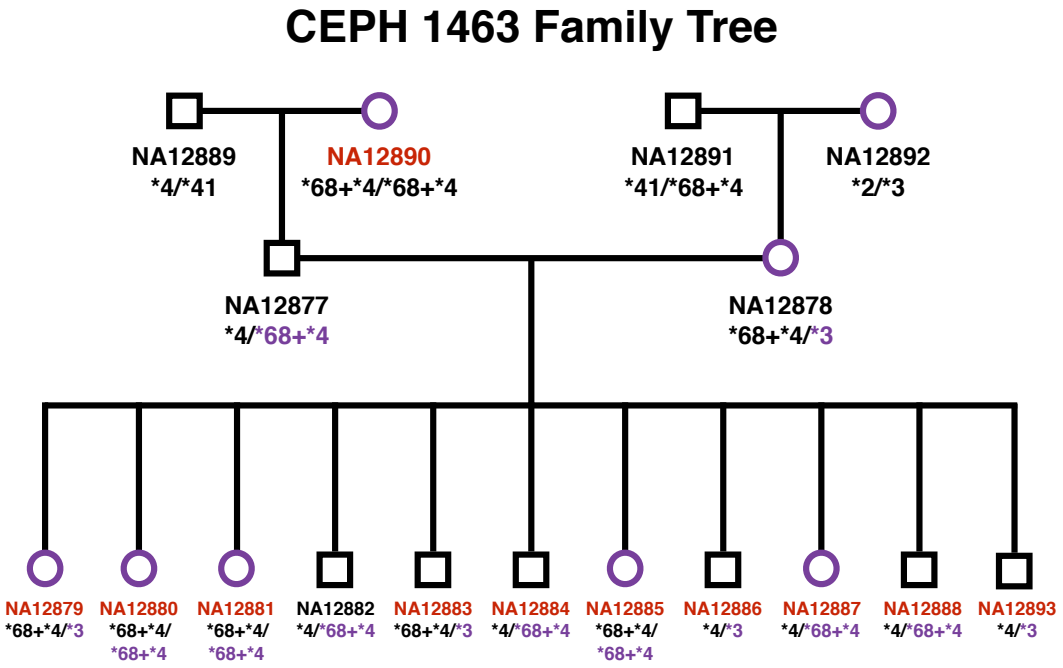
Sample ID	Family	Ethnicity	Gender	*CYP2A6* genotype	Cypiripi^{++} prediction
HG01190	PR40	Puerto Rican	F	*1/*1	✓ (*1/*1)
NA07357	1345	European	F	*1/*1	✓ (*1/*1)
NA07348	1345	European	C	*1/*1	✓ (*1/*1)
NA10854	1349	European	M	*1/*1	✓ (*1/*1)
NA12003	1420	European	F	*1/*1 or *1/*8	✓ (*1/*1)
NA12156	1408	European	M	*1/*1	✓ (*1/*1)
NA10831	1408	European	C	*1/*2	✓ (*1/*2)
NA12878	1463	European	C	*1/*1	✓ (*1/*1)
NA19239	Y117	Yoruban	F	*1/*17	✓ (*1/*17)
NA19789	M037	Mexican Am	F	*1/*1	✓ (*1/*1)
NA19819	2418	AA	M	*1/*1	✓ (*1/*1)

For case (1), *35 allele is called as *2 for the samples NA10861, NA11984 and NA12003. However, as reported by [142], NA12003 actually contains *35 allele instead of *2, and this discrepancy is mostly due to the inability of TaqMan assays to properly genotype *35 allele [151]. The similar happens in case (4) with samples NA19834, NA19835 and NA19836, where we have SNP c.1716 G>A. This SNP differentiates alleles *2 and *45 and it is not present in TaqMan assays [45]. Case (4) presents *15, another allele problematic for most of the current genotyping platforms. For example, TaqMan assays often confuse it with other alleles [151]. One of the reasons for this is c.137 insT, which defines *15 allele but is also present in all *13 fusion alleles. However, [142] confirmed that NA19239's actual genotype contains *15, which matches our results as well. In case (2), copy number results for *13-like fusion allele *76 are not clear for samples NA19200 and NA19202. However, additional validation by [45] confirms that *76 is not present, which matches Cypiripi^{++}'s prediction. Furthermore, we could not find any evidence of increased coverage in *CYP2D7* region, that would be suggested by the existence of *76. All those cases suggest that Cypiripi^{++} provides more accurate genotyping results than currently used genotyping panels, especially in the presence of recently discovered alleles. In the case of NA10860, Cypiripi detects *4 allele duplication, which PCR-based methods miss. We have cross-validated our prediction by running Cypiripi^{++} on Illumina HiSeq X WGS NA10860 sample publicly available from https://export.uppmax.uu.se/a2009002/opendata/HiSeqX_CEPH/. This sample was sequenced with approx. 28× depth of coverage (or 14× per chromosome). By simple coverage analysis, it is clear that there are at least 3 copies of *CYP2D6*, since average coverage of *CYP2D6* region is 42×. Thus, we assume that the correct allele is indeed *1/*4+*4.

We have not observed any disagreements with Mendelian laws of inheritance when using Cypiripi^{++} on PGRNseq data. This is in sharp contrast with previous PGRNseq data analysis which relied on SNP callers to infer genotypes [59].

When it comes to Illumina WGS data, genotypes predicted by Cypiripi^{++} are in concordance with genotypes validated in [176, 45, 142], as shown in Table 7.2. Although we do not have genotype information for some members of CEPH 1463 family, we show that our predictions are in full accordance with Mendelian laws of inheritance, as depicted in Figure 7.3.

Figure 7.3: CEPH 1463 Family Tree with the Cypiripi^{++} genotype predictions for *CYP2D6*. Purple alleles indicate alleles inherited from mother, while black alleles indicate alleles inherited from father side. Red sample IDs indicate the lack of validation. As can be seen, all genotypes follow Mendelian laws of inheritance.



We also include our predictions for *CYP2A6* genotype on samples for which the validation is available. These results are shown in Table 7.3. As it can be seen, Cypiripi^{++} provides an accurate *CYP2A6* genotype calls for all of the samples. Genotype validation for these cases is available in [142].

In addition to its genotyping accuracy, Cypiripi^{++} has very low computational overhead. In our experiments, each run required less than 10 seconds and fewer than 100 MB of memory even for the high-coverage PGRNseq samples.

7.2.3 Novel alleles

In many samples we have observed that sub-alleles detected by Cypiripi^{++} are not present in the online database. Similar observation was made by [147] regarding the *CYP2D6*2* family of sub-alleles. For example, c.843 T>G, associated with all recently discovered *4 sub-alleles (e.g. *4M, *4N and *4P), is not associated with *4 alleles (e.g. *4A, *4B etc.) discovered earlier. However, we have found multiple samples where the evidence strongly suggests that *4A allele contains this SNP. This implies either the incomplete characterization of *4A sub-allele, or the presence of novel *4 sub-alleles.

Since the lack of non-functional SNPs can affect the accurate genotype interpretation of HTS-based tools (as already reported by [176]), Cypiripi^{++} ships with the updated database which contains additional sub-alleles believed to exist in the wild. Further studies are needed for complete characterization of those sub-alleles.

7.3 Conclusion

In this chapter, we have presented the first computational tool which is able to accurately infer genotypes of *CYP2D6*, *CYP2A6* and most of the other ADMER genes from PGRNseq data. This tool, dubbed Cypiripi^{++}, also supports Illumina WGS data and detects various structural rearrangements occurring within the target gene regions. Fast execution and low system requirements make Cypiripi^{++} highly suitable for clinical settings where speed is of high importance.

There are still some challenges which need to be addressed in the future work. One of them is exact characterization of novel alleles and sub-alleles, which is not yet possible with current HTS technologies. However, this problem can be resolved with the use of platforms which can provide long read lengths and additional metadata (e.g. haplotype information). Other challenges to be resolved are integration of the error model which describes the variance of coverage depth in the copy number estimation model, and detection of fusion breakpoints in the completely identical regions. Currently, Cypiripi^{++} is not able to precisely locate the fusion breakpoints in such regions (although it is able to detect the presence of such fusion events).

Specialized HTS platforms, such are PGRNseq, are removing the last obstacles preventing the wider integration of HTS technologies in everyday clinical settings. Coupled with fast and accurate genotyping frameworks, such are Cypiripi^{++}, these platforms can assist physicians in tailoring the prescription recommendations based on patient's genetic makeup and eventually lead to improved medical care.

Chapter 8

Conclusion

High throughput sequencing (HTS) platforms have made significant contributions to various fields of genomic research. For these reasons, HTS technologies are starting to make foray into clinical environments. One problem of particular clinical significance is the precise genotype inference of pharmaceutically important genes. Specialized HTS technologies, like PGRNseq, promise fast and cost-effective solutions for such problem. However, challenges like the data storage and transfer, as well as the accurate data analysis of HTS data are still preventing the wider integration of HTS platforms in clinical settings.

This thesis presents fast and efficient methods for HTS data compression, which significantly improve over currently available compression schemes. It also introduces a first computational method for clinical genotyping which uses HTS data to infer a genotype. The ultimate goal of these methods is to ease the adoption of HTS technologies in clinical environments.

First we have introduced DeeZ, a novel HTS data compression tool for read alignments stored in SAM/BAM file formats. DeeZ uses local assembly to detect common variants which are redundantly encoded by many reads, and significantly improves the compression rate by encoding such variations only once. We have shown that DeeZ consistently provides the best compression rates among evaluated tools, achieving up to 50% improvement over commonly used BAM file format. Furthermore, DeeZ is among fastest compression tools, and unlike many other tools, it provides a random access to any record within the compressed archive.

Then we presented a comprehensive framework designed for benchmarking the performance of various HTS compression tools. This framework has been developed as a part of Moving Picture Experts Group (MPEG) activity to explore the current landscape of genomic data compression algorithms, and to design an open standard for storage and transport of such data. The performance of more than 25 compression tools for both FASTQ and SAM file formats have been evaluated on the large set of publicly available samples. These samples were carefully selected by MPEG community to cover the wide

spectrum of technologies and species necessary for ensuring statistically meaningful results. We have shown that significant improvements in both compression time and performance can be made over the commonly used Gzip and BAM file formats. This study also provided a further evidence that DeeZ offers the best compression rates for majority of SAM/BAM files, making it the most optimal choice for SAM/BAM compression and archival.

In the second part of this thesis, we have discussed the accurate genotype inference for pharmaceutically important genes (also known as ADMER genes), with the particular emphasis on *CYP2D6* and *CYP2A6*. These two genes are highlighted because they play significant role in metabolism of more than 25% clinically important drugs, as well as smoking habits. They also harbour various structural variations with the neighbouring and highly homologous pseudogenes, which hinders the performance of various genotyping panels. We show that HTS data can be used to infer the accurate genotype of *CYP2D6* by introducing Cypiripi, a first computational tool which can obtain *CYP2D6* genotypes from the HTS data with uniform coverage. Cypiripi models the genotyping problem as an instance of Integer Linear Programming. This model is able to detect copy number aberrations, deletions and fusions with closely related pseudogene *CYP2D7*. We demonstrate the performance of Cypiripi on large set of simulations designed to cover the majority of alleles observed in the literature, and on the real data set covering the single family. Cypiripi's performance met our expectations by successfully estimating the genotype on all samples.

Finally, we relax the constraint that the coverage must be uniform, and show that Cypiripi can be extended to work with specialized HTS technologies like PGRNseq which target specific set of pharmacogenes with non-uniform coverage. The new tool, called Cypiripi^{++}, also improves the detection of various fusion alleles, and brings down the overall running time and memory requirements. Cypiripi^{++} also brings the support for *CYP2A6* gene. We have successfully used Cypiripi^{++} to identify correct genotypes of 96 subjects sequenced with PGRNseq. Furthermore, we used Cypiripi^{++} to genotype the whole extended CEPH 1463 family, where all of observed genotypes were in concordance with Mendelian laws of inheritance and previously reported validations.

8.1 Future Work

Recent focus on longer reads and higher depths of coverage in HTS technologies indicates that the amount of generated data will grow even further in the near future. Thus, more efficient compression algorithms which can deal with long reads will be necessary to accommodate such data. This is especially important because most long read technologies come with high error rates, which severely affect the majority of compression schemes used for sequence compression. Another major obstacle faced by current compression methods is high entropy of auxiliary data (e.g. quality scores or read identifiers), which currently accounts for a major portion of compressed data regardless of the method being used. This problem

will require a study of various lossy compression schemes which can significantly decrease the entropy of such data while minimizing the impact on the downstream analysis. Such study is currently being conducted by MPEG as a part of Core Experiments on Genomic Information.

Development of longer reads and barcoded sequencing also provides an opportunity for the more accurate ADMER genotyping. Natural future directions would consist of utilizing longer reads and haplotype annotations for identifying novel alleles, resolving potential ambiguities in genotype inference, and more precise discovery of various structural variations in regions of pharmaceutical importance. Moreover, integration of advanced coverage bias models should be investigated for more fine-grained ambiguity resolution during the genotyping process. Finally, the planned addition of wider set of ADMER genes to our current pipeline (such are *CYP2C9*, *CYP2B6*, *TMPT* and *VKORC1*) will make Cypiripi a single easy-to-use and cost-effective platform for general ADMER genotyping, which can ultimately boost or even replace the various genotyping panels used for aiding drug prescription and treatment decisions.

Bibliography

[1] An integrated map of genetic variation from 1,092 human genomes. *Nature 491*, 7422 (Nov. 2012), 56–65.

[2] President Obama's Precision Medicine Initiative. *whitehouse.gov* (2015-01-30). https://www.whitehouse.gov/the-press-office/2015/01/30/fact-sheet-president-obama-s-precision-medicine-initiative.

[3] 1000 Genomes Project Consortium, Abecasis, G. R., Altshuler, D., Auton, A., Brooks, L. D., Durbin, R. M., Gibbs, R. A., Hurles, M. E., and McVean, G. A. A Map of Human Genome Variation from Population-Scale Sequencing. *Nature 467*, 7319 (Oct. 2010), 1061–1073.

[4] Alkan, C., Kidd, J. M., Marques-Bonet, T., Aksay, G., Antonacci, F., Hormozdiari, F., Kitzman, J. O., Baker, C., Malig, M., Mutlu, O., Sahinalp, S. C., Gibbs, R. A., and Eichler, E. E. Personalized Copy-Number and Segmental Duplication Maps using Next-Generation Sequencing. *Nature Genetics 41*, 10 (Oct. 2009), 1061–1067.

[5] Annala, M., Parker, B., Zhang, W., and Nykter, M. Fusion Genes and their Discovery Using High Throughput Sequencing. *Cancer Letters 340*, 2 (Nov. 2013).

[6] Antonio Diaz Diaz. Lzip - LZMA lossless data compressor. http://www.nongnu.org/lzip/lzip.html.

[7] Ashley, E. A. Towards Precision Medicine. *Nature Reviews Genetics 17*, 9 (Sept. 2016), 507–522.

[8] Ashley, E. A., Butte, A. J., Wheeler, M. T., Chen, R., Klein, T. E., Dewey, F. E., Dudley, J. T., Ormond, K. E., Pavlovic, A., Hudgins, L., Gong, L., Hodges, L. M., Berlin, D. S., Thorn, C. F., Sangkuhl, K., Hebert, J. M., Woon, M., Sagreiya, H., Whaley, R., Morgan, A. A., Pushkarev, D., Neff, N. F., Knowles, J. W., Chou, M., Thakuria, J., Rosenbaum, A., Zaranek, A. W., Church, G., Greely, H. T., Quake, S. R., and Altman, R. B. Clinical Evaluation Incorporating a Personal Genome. *Lancet 375*, 9725 (May 2010), 1525–1535.

[9] Bao, R., Huang, L., Andrade, J., Tan, W., Kibbe, W. A., Jiang, H., and Feng, G. Review of Current Methods, Applications, and Data Management for the Bioinformatics Analysis of Whole Exome Sequencing. *Cancer Informatics 13*, Suppl 2 (Sept. 2014), 67–82.

[10] Bauer, M. J., Cox, A. J., and Rosone, G. Lightweight BWT construction for very large string collections. In *Combinatorial Pattern Matching*. Springer, 2011, pp. 219–231.

[11] Bell, P. A., Chaturvedi, S., Gelfand, C. A., Huang, C. Y., Kochersperger, M., Kopla, R., Modica, F., Pohl, M., Varde, S., Zhao, R., Zhao, X., Boyce-Jacino, M. T., and Yassen, A. SNPstream UHT: Ultra-high throughput SNP genotyping for pharmacogenomics and drug discovery. *BioTechniques Suppl* (June 2002), 70–72, 74, 76–77.

[12] Benoit, G., Lemaitre, C., Lavenier, D., Drezen, E., Dayris, T., Uricaru, R., and Rizk, G. Reference-free compression of high throughput sequencing data with a probabilistic de Bruijn graph. *BMC Bioinformatics 16* (2015), 288.

[13] Bentley, D. R., Balasubramanian, S., Swerdlow, H. P., Smith, G. P., Milton, J., Brown, C. G., Hall, K. P., Evers, D. J., Barnes, C. L., Bignell, H. R., Boutell, J. M., Bryant, J., Carter, R. J., Keira Cheetham, R., Cox, A. J., Ellis, D. J., Flatbush, M. R., Gormley, N. A., Humphray, S. J., Irving, L. J., Karbelashvili, M. S., Kirk, S. M., Li, H., Liu, X., Maisinger, K. S., Murray, L. J., Obradovic, B., Ost, T., Parkinson, M. L., Pratt, M. R., Rasolonjatovo, I. M. J., Reed, M. T., Rigatti, R., Rodighiero, C., Ross, M. T., Sabot, A., Sankar, S. V., Scally, A., Schroth, G. P., Smith, M. E., Smith, V. P., Spiridou, A., Torrance, P. E., Tzonev, S. S., Vermaas, E. H., Walter, K., Wu, X., Zhang, L., Alam, M. D., Anastasi, C., Aniebo, I. C., Bailey, D. M. D., Bancarz, I. R., Banerjee, S., Barbour, S. G., Baybayan, P. A., Benoit, V. A., Benson, K. F., Bevis, C., Black, P. J., Boodhun, A., Brennan, J. S., Bridgham, J. A., Brown, R. C., Brown, A. A., Buermann, D. H., Bundu, A. A., Burrows, J. C., Carter, N. P., Castillo, N., Chiara E. Catenazzi, M., Chang, S., Neil Cooley, R., Crake, N. R., Dada, O. O., Diakoumakos, K. D., Dominguez-Fernandez, B., Earnshaw, D. J., Egbujor, U. C., Elmore, D. W., Etchin, S. S., Ewan, M. R., Fedurco, M., Fraser, L. J., Fuentes Fajardo, K. V., Scott Furey, W., George, D., Gietzen, K. J., Goddard, C. P., Golda, G. S., Granieri, P. A., Green, D. E., Gustafson, D. L., Hansen, N. F., Harnish, K., Haudenschild, C. D., Heyer, N. I., Hims, M. M., Ho, J. T., Horgan, A. M., Hoschler, K., Hurwitz, S., Ivanov, D. V., Johnson, M. Q., James, T., Huw Jones, T. A., Kang, G.-D., Kerelska, T. H., Kersey, A. D., Khrebtukova, I., Kindwall, A. P., Kingsbury, Z., Kokko-Gonzales, P. I., Kumar, A., Laurent, M. A., Lawley, C. T., Lee, S. E., Lee, X., Liao, A. K., Loch, J. A., Lok, M., Luo, S., Mammen, R. M., Martin, J. W., McCauley, P. G., McNitt, P., Mehta, P., Moon, K. W., Mullens, J. W., Newington, T., Ning, Z., Ling Ng, B., Novo, S. M., O'Neill, M. J., Osborne, M. A., Osnowski, A., Ostadan, O., Paraschos, L. L., Pickering, L., Pike, A. C., Pike, A. C., Chris Pinkard, D., Pliskin, D. P., Podhasky, J., Quijano, V. J., Raczy, C., Rae, V. H., Rawlings, S. R., Chiva Rodriguez, A., Roe, P. M., Rogers, J., Rogert Bacigalupo, M. C., Romanov, N., Romieu, A., Roth, R. K., Rourke, N. J., Ruediger, S. T., Rusman, E., Sanches-Kuiper, R. M., Schenker, M. R., Seoane, J. M., Shaw, R. J., Shiver, M. K., Short,

S. W., Sizto, N. L., Sluis, J. P., Smith, M. A., Ernest Sohna Sohna, J., Spence, E. J., Stevens, K., Sutton, N., Szajkowski, L., Tregidgo, C. L., Turcatti, G., vandeVondele, S., Verhovsky, Y., Virk, S. M., Wakelin, S., Walcott, G. C., Wang, J., Worsley, G. J., Yan, J., Yau, L., Zuerlein, M., Rogers, J., Mullikin, J. C., Hurles, M. E., McCooke, N. J., West, J. S., Oaks, F. L., Lundberg, P. L., Klenerman, D., Durbin, R., and Smith, A. J. Accurate Whole Human Genome Sequencing Using Reversible Terminator Chemistry. *Nature 456*, 7218 (Nov. 2008), 53–59.

[14] Bentley, J. L., Sleator, D. D., Tarjan, R. E., and Wei, V. K. A Locally Adaptive Data Compression Scheme. *Commun. ACM 29*, 4 (Apr. 1986), 320–330.

[15] Bonfield, J. K. The Scramble conversion tool. *Bioinformatics* (2014).

[16] Bonfield, J. K., and Mahoney, M. V. Compression of FASTQ and SAM Format Sequencing Data. *PLoS ONE 8*, 3 (2013), e59190.

[17] Broad Institute. Picard Tools. http://broadinstitute.github.io/picard/.

[18] Burrows, M., and Wheeler, D. J. A block-sorting lossless data compression algorithm. http://www.hpl.hp.com/techreports/Compaq-DEC/SRC-RR-124.pdf.

[19] Buxbaum, J. D., Daly, M. J., Devlin, B., Lehner, T., Roeder, K., and State, M. W. The Autism Sequencing Consortium: Large-Scale, High-Throughput Sequencing in Autism Spectrum Disorders. *Neuron 76*, 6 (Dec. 2012), 1052–1056.

[20] Campagne, F., Dorff, K. C., Chambwe, N., Robinson, J. T., and Mesirov, J. P. Compression of Structured High-Throughput Sequencing Data. *PLoS ONE 8*, 11 (Nov. 2013), e79871.

[21] Cavallari, L. H. Tailoring drug therapy based on genotype. *Journal of Pharmacy Practice 25*, 4 (2012), 413–416.

[22] Chaisson, M. J., Huddleston, J., Dennis, M. Y., Sudmant, P. H., Malig, M., Hormozdiari, F., Antonacci, F., Surti, U., Sandstrom, R., Boitano, M., Landolin, J. M., Stamatoyannopoulos, J. A., Hunkapiller, M. W., Korlach, J., and Eichler, E. E. Resolving the Complexity of the Human Genome Using Single-Molecule Sequencing. *Nature 517*, 7536 (Jan. 2015), 608–611.

[23] Check Hayden, E. Technology: The $1,000 genome. *Nature 507*, 7492 (Mar. 2014), 294–295.

[24] Cleary, J., and Witten, I. Data Compression Using Adaptive Coding and Partial String Matching. *IEEE Transactions on Communications 32*, 4 (Apr. 1984), 396–402.

[25] Cock, P. J. A., Fields, C. J., Goto, N., Heuer, M. L., and Rice, P. M. The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. *Nucleic Acids Research 38*, 6 (Apr. 2010), 1767–1771.

[26] Collins, F. S., and Varmus, H. A New Initiative on Precision Medicine. *New England Journal of Medicine 372*, 9 (Feb. 2015), 793–795.

[27] Cox, A. J., Bauer, M. J., Jakobi, T., and Rosone, G. Large-scale compression of genomic sequence databases with the Burrows-Wheeler transform. *Bioinformatics 28*, 11 (June 2012), 1415–1419.

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

[29] Dao, P., Numanagić, I., Lin, Y.-Y., Hach, F., Karakoc, E., Donmez, N., Collins, C., Eichler, E. E., and Sahinalp, S. C. ORMAN: Optimal Resolution of Ambiguous RNA-Seq Multimappings in the Presence of Novel Isoforms. *Bioinformatics 30*, 5 (Jan. 2014), 644–651.

[30] Deorowicz, S., and Grabowski, S. Compression of DNA sequence reads in FASTQ format. *Bioinformatics 27*, 6 (2011), 860–862.

[31] Deorowicz, S., and Grabowski, S. Data compression for sequencing data. *Algorithms for Molecular Biology 8*, 1 (2013), 1–13.

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

[33] Dewey, F. E., Grove, M. E., Pan, C., Goldstein, B. A., Bernstein, J. A., Chaib, H., Merker, J. D., Goldfeder, R. L., Enns, G. M., David, S. P., Pakdaman, N., Ormond, K. E., Caleshu, C., Kingham, K., Klein, T. E., Whirl-Carrillo, M., Sakamoto, K., Wheeler, M. T., Butte, A. J., Ford, J. M., Boxer, L., Ioannidis, J. P. A., Yeung, A. C., Altman, R. B., Assimes, T. L., Snyder, M., Ashley, E. A., and Quertermous, T. Clinical Interpretation and Implications of Whole-Genome Sequencing. *JAMA 311*, 10 (Mar. 2014), 1035–1045.

[34] Djebali, S., Davis, C. A., Merkel, A., Dobin, A., Lassmann, T., Mortazavi, A. M., Tanzer, A., Lagarde, J., Lin, W., Schlesinger, F., Xue, C., Marinov, G. K., Khatun, J., Williams, B. A., Zaleski, C., Rozowsky, J., Röder, M., Kokocinski, F., Abdelhamid, R. F., Alioto, T., Antoshechkin, I., Baer, M. T., Bar, N. S., Batut, P., Bell, K., Bell, I., Chakrabortty, S., Chen, X., Chrast, J., Curado, J., Derrien, T., Drenkow, J., Dumais, E., Dumais, J., Duttagupta, R., Falconnet, E., Fastuca, M., Fejes-Toth, K., Ferreira, P., Foissac, S., Fullwood, M. J., Gao, H., Gonzalez, D., Gordon, A., Gunawardena, H., Howald, C., Jha, S., Johnson, R., Kapranov, P., King, B., Kingswood, C., Luo, O. J., Park, E., Persaud, K., Preall, J. B., Ribeca, P., Risk, B., Robyr, D., Sammeth, M., Schaffer, L., See, L.-H., Shahab, A., Skancke, J., Suzuki, A. M., Takahashi, H., Tilgner, H., Trout, D., Walters, N., Wang, H., Wrobel, J., Yu, Y., Ruan, X., Hayashizaki, Y., Harrow, J., Gerstein, M., Hubbard, T., Reymond, A.,

Antonarakis, S. E., Hannon, G., Giddings, M. C., Ruan, Y., Wold, B., Carninci, P., Guigó, R., and Gingeras, T. R. Landscape of Transcription in Human Cells. *Nature 489*, 7414 (Sept. 2012), 101–108.

[35] Dohm, J. C., Lottaz, C., Borodina, T., and Himmelbauer, H. Substantial Biases in Ultra-Short Read Data Sets from High-Throughput DNA Sequencing. *Nucleic Acids Research 36*, 16 (Sept. 2008), e105.

[36] Dong, L., Wang, W., Li, A., Kansal, R., Chen, Y., Chen, H., and Li, X. Clinical Next Generation Sequencing for Precision Medicine in Cancer. *Current Genomics 16*, 4 (Aug. 2015), 253–263.

[37] Duda, J. Asymmetric numeral systems: Entropy coding combining speed of Huffman coding with compression rate of arithmetic coding. http://arxiv.org/abs/1311.2540.

[38] Dutta, A., Haque, M. M., Bose, T., Reddy, C. V. S. K., and Mande, S. S. FQC: A novel approach for efficient compression, archival, and dissemination of fastq datasets. *Journal of Bioinformatics and Computational Biology 13*, 3 (June 2015), 1541003.

[39] Eid, J., Fehr, A., Gray, J., Luong, K., John Lyle, Otto, G., Peluso, P., Rank, D., Baybayan, P., Bettman, B., Bibillo, A., Bjornson, K., Bidhan Chaudhuri, Christians, F., Cicero, R., Clark, S., Dalal, R., Dewinter, A., Dixon, J., Foquet, M., Gaertner, A., Hardenbol, P., Heiner, C., Kevin Hester, Holden, D., Kearns, G., Kong, X., Ronald Kuse, Lacroix, Y., Lin, S., Lundquist, P., Congcong Ma, Marks, P., Maxham, M., Murphy, D., Park, I., Pham, T., Phillips, M., Roy, J., Sebra, R., Shen, G., Sorenson, J., Tomaney, A., Travers, K., Trulson, M., Vieceli, J., Wegener, J., Wu, D., Yang, A., Zaccarin, D., Zhao, P., Zhong, F., Korlach, J., and Turner, S. Real-time DNA sequencing from single polymerase molecules. *Science 323*, 5910 (Jan. 2009), 133–138.

[40] Elias, P. Universal codeword sets and representations of the integers. *IEEE Transactions on Information Theory 21*, 2 (Mar. 1975), 194–203.

[41] ENCODE Project Consortium. An Integrated Encyclopedia of DNA Elements in the Human Genome. *Nature 489*, 7414 (Sept. 2012), 57–74.

[42] Ewing, B., and Green, P. Base-calling of automated sequencer traces using phred. II. Error probabilities. *Genome Research 8*, 3 (Mar. 1998), 186–194.

[43] Ezra, J. GitHub - Infinidat/slimfastq: Fast, efficient, lossless compression of fastq files. https://github.com/Infinidat/slimfastq.

[44] Fan, L., Cao, P., Almeida, J., and Broder, A. Z. Summary Cache: A Scalable Wide-area Web Cache Sharing Protocol. *IEEE/ACM Trans. Netw. 8*, 3 (June 2000), 281–293.

[45] Fang, H., Liu, X., Ramírez, J., Choudhury, N., Kubo, M., Im, H., Konkashbaev, A., Cox, N., Ratain, M., Nakamura, Y., and others. Establishment of

CYP2D6 reference samples by multiple validated genotyping platforms. *The Pharmacogenomics Journal 14*, 6 (2014), 564–572.

[46] Fernandez-Salguero, P., Hoffman, S. M., Cholerton, S., Mohrenweiser, H., Raunio, H., Rautio, A., Pelkonen, O., Huang, J. D., Evans, W. E., and Idle, J. R. A Genetic Polymorphism in Coumarin 7-Hydroxylation: Sequence of the Human CYP2A Genes and Identification of Variant CYP2A6 Alleles. *American Journal of Human Genetics 57*, 3 (Sept. 1995), 651–660.

[47] Frackiewicz, E. J., Shiovitz, T. M., and Jhee, S. S. *Ethnicity in Drug Development and Therapeutics*. Cambridge University Press, Page 37, June 2011.

[48] Gaedigk, A. Complexities of CYP2D6 gene analysis and interpretation. *International Review of Psychiatry 25*, 5 (Oct. 2013), 534–553.

[49] Gaedigk, A., Jaime, L. K. M., Bertino Jr, J. S., Bérard, A., Pratt, V. M., Bradfordand, L. D., and Leeder, J. S. Identification of novel CYP2D7-2D6 hybrids: Non-functional and functional variants. *Frontiers in Pharmacology 1* (2010).

[50] Gaedigk, A., Simon, S., Pearce, R., Bradford, L., Kennedy, M., and Leeder, J. The CYP2D6 activity score: Translating genotype information into a qualitative measure of phenotype. *Clinical Pharmacology & Therapeutics 83*, 2 (2007), 234–242.

[51] Gailly, J.-L., and Deutsch, P. ZLIB Compressed Data Format Specification version 3.3. https://tools.ietf.org/html/rfc1950.

[52] Giancarlo, R., Rombo, S. E., and Utro, F. Compressive biological sequence analysis and archival in the era of high-throughput sequencing technologies. *Briefings in Bioinformatics 15*, 3 (2014), 390–406.

[53] Giesen, F. Interleaved entropy coders. http://arxiv.org/abs/1402.3392.

[54] Gnerre, S., Maccallum, I., Przybylski, D., Ribeiro, F. J., Burton, J. N., Walker, B. J., Sharpe, T., Hall, G., Shea, T. P., Sykes, S., Berlin, A. M., Aird, D., Costello, M., Daza, R., Williams, L., Nicol, R., Gnirke, A., Nusbaum, C., Lander, E. S., and Jaffe, D. B. High-Quality Draft Assemblies of Mammalian Genomes from Massively Parallel Sequence Data. *Proceedings of the National Academy of Sciences of the United States of America 108*, 4 (Jan. 2011), 1513–1518.

[55] Goldberg, B., Sichtig, H., Geyer, C., Ledeboer, N., and Weinstock, G. M. Making the Leap from Research Laboratory to Clinic: Challenges and Opportunities for Next-Generation Sequencing in Infectious Disease Diagnostics. *mBio 6*, 6 (Dec. 2015).

[56] Goldfeder, R. L., Priest, J. R., Zook, J. M., Grove, E. M., Waggott, D., Wheeler, M. T., Salit, M., and Ashley, A. E. Medical Implications of Technical Accuracy in Genome Sequencing. *Genome Medicine 8* (2016), 24.

[57] Golomb, S. Run-length encodings (Corresp.). *IEEE Transactions on Information Theory 12*, 3 (July 1966), 399–401.

[58] Goodwin, S., McPherson, J. D., and McCombie, W. R. Coming of age: Ten years of next-generation sequencing technologies. *Nature Reviews Genetics 17*, 6 (June 2016), 333–351.

[59] Gordon, A. S., Fulton, R. S., Qin, X., Mardis, E. R., Nickerson, D. A., and Scherer, S. PGRNseq: A Targeted Capture Sequencing Panel for Pharmacogenetic Research and Implementation. *Pharmacogenetics and Genomics* (Jan. 2016).

[60] Grabowski, S., Deorowicz, S., and Roguski, L. Disk-based compression of data from genome sequencing. *Bioinformatics 31*, 9 (May 2015), 1389–1395.

[61] Green, R. C., Rehm, H. L., and Kohane, I. S. Chapter 9 - Clinical Genome Sequencing A2 - Ginsburg, Geoffrey S. In *Genomic and Personalized Medicine (Second Edition)*, H. F. Willard, Ed. Academic Press, 2013, pp. 102–122.

[62] Greenwood, A. D., and Burke, D. T. Single nucleotide primer extension: Quantitative range, variability, and multiplex analysis. *Genome Research 6*, 4 (Jan. 1996), 336–348.

[63] Gu, D. F., Hinks, L. J., Morton, N. E., and Day, I. N. The use of long PCR to confirm three common alleles at the CYP2A6 locus and the relationship between genotype and smoking habit. *Annals of Human Genetics 64*, Pt 5 (Sept. 2000), 383–390.

[64] Hach, F. *Scalable Mapping and Compression of High Throughput Genome Sequencing Data*. Thesis, Applied Sciences: School of Computing Science, July 2013.

[65] Hach, F., Hormozdiari, F., Alkan, C., Hormozdiari, F., Birol, I., Eichler, E. E., and Sahinalp, S. C. mrsFAST: A cache-oblivious algorithm for short-read mapping. *Nature Methods 7*, 8 (2010), 576–577.

[66] Hach, F., Numanagić, I., Alkan, C., and Sahinalp, S. C. SCALCE: Boosting Sequence Compression Algorithms Using Locally Consistent Encoding. *Bioinformatics 28*, 23 (Jan. 2012), 3051–3057.

[67] Hach, F., Numanagić, I., and Sahinalp, S. C. DeeZ: Reference-Based Compression by Local Assembly. *Nature Methods 11*, 11 (Nov. 2014), 1082–1084.

[68] Hach, F., Sarrafi, I., Hormozdiari, F., Alkan, C., Eichler, E. E., and Sahinalp, S. C. mrsFAST-Ultra: A compact, SNP-aware mapper for high performance sequencing applications. *Nucleic Acids Research 42*, W1 (2014), W494–W500.

[69] Haussler, D., O'Brien, S. J., Ryder, O. A., Barker, F. K., Clamp, M., Crawford, A. J., Hanner, R., Hanotte, O., Johnson, W. E., McGuire, J. A., Miller, W., Murphy, R. W., Murphy, W. J., Sheldon, F. H., Sinervo, B., Venkatesh, B., Wiley, E. O., Allendorf, F. W., Amato, G., Baker, C. S., Bauer, A., Beja-Pereira, A., Bermingham, E., Bernardi, G., Bonvicino, C. R., Brenner, S., Burke, T., Cracraft, J., Diekhans, M., Edwards, S., Ericson, P. G., Estes, J., Fjelsda, J., Flesness, N., Gamble, T., Gaubert, P., Graphodatsky, A. S., Marshall Graves, J. A., Green, E. D., Green, R. E., Hackett, S., Hebert, P., Helgen, K. M., Joseph, L.,

Kessing, B., Kingsley, D. M., Lewin, H. A., Luikart, G., Martelli, P., Moreira, M. A., Nguyen, N., Orti, G., Pike, B. L., Rawson, D. M., Schuster, S. C., Seuanez, H. N., Shaffer, H. B., Springer, M. S., Stuart, J. M., Sumner, J., Teeling, E., Vrijenhoek, R. C., Ward, R. D., Warren, W. C., Wayne, R., Williams, T. M., Wolfe, N. D., and Zhang, Y. P. Genome 10K: A Proposal to Obtain Whole-Genome Sequence for 10 000 Vertebrate Species. *J. Hered. 100* (2009), 659–674.

[70] Ho, M. K., and Tyndale, R. F. Overview of the pharmacogenomics of cigarette smoking. *The Pharmacogenomics Journal 7*, 2 (Jan. 2007), 81–98.

[71] Holland, P. M., Abramson, R. D., Watson, R., and Gelfand, D. H. Detection of specific polymerase chain reaction product by utilizing the 5'—-3' exonuclease activity of Thermus aquaticus DNA polymerase. *Proceedings of the National Academy of Sciences of the United States of America 88*, 16 (Aug. 1991), 7276–7280.

[72] Holland, R. C., and Lynch, N. Sequence squeeze: An open contest for sequence compression. *GigaScience 2* (Apr. 2013), 5.

[73] Hormozdiari, F., Hajirasouliha, I., Dao, P., Hach, F., Yorukoglu, D., Alkan, C., Eichler, E. E., and Sahinalp, S. C. Next-Generation VariationHunter: Combinatorial Algorithms for Transposon Insertion Discovery. *Bioinformatics 26*, 12 (June 2010), i350–i357.

[74] Horn, J. R., and Hansten, P. D. Get to Know an Enzyme: CYP2D6. *Pharmacy Times* (Tuesday, July 1, 2008). http://www.pharmacytimes.com/publications/issue/2008/2008-07/2008-07-8624.

[75] Hsi-Yang Fritz, M., Leinonen, R., Cochrane, G., and Birney, E. Efficient storage of high throughput DNA sequencing data using reference-based compression. *Genome Research 21*, 5 (May 2011), 734–740.

[76] Huang, F. W., Hodis, E., Xu, M. J., Kryukov, G. V., Chin, L., and Garraway, L. A. Highly Recurrent TERT Promoter Mutations in Human Melanoma. *Science (New York, N.Y.) 339*, 6122 (Feb. 2013), 957–959.

[77] Huffman, D. A Method for the Construction of Minimum-Redundancy Codes. *Proceedings of the IRE 40*, 9 (Sept. 1952), 1098–1101.

[78] Hukkanen, J., Jacob, P., and Benowitz, N. L. Metabolism and Disposition Kinetics of Nicotine. *Pharmacological Reviews 57*, 1 (Mar. 2005), 79–115.

[79] Human Microbiome Project Consortium. Structure, Function and Diversity of the Healthy Human Microbiome. *Nature 486*, 7402 (June 2012), 207–214.

[80] Illumina Inc. HiSeq X^{TM}Series of Sequencing Systems. http://www.illumina.com/documents/products/datasheets/datasheet-hiseq-x-ten.pdf.

[81] Ingelman-Sundberg, M. Genetic polymorphisms of cytochrome P450 2D6 (CYP2D6): Clinical consequences, evolutionary aspects and functional diversity. *The Pharmacogenomics Journal 5*, 1 (2004), 6–13.

[82] ISO/IEC SC29 WG11 (MPEG) – Requirements. N15346 - Investigation on genomic information compression and storage. (Geneva).

[83] ISO/IEC SC29 WG11 (MPEG) – Requirements. N15740 - Call for Evidence (CfE) for Genome Compression and Storage. (Geneva).

[84] ISO/IEC SC29 WG11 (MPEG) – Requirements. N16145 - Database for Evaluation of Genomic Information Representation and Compression. (San Diego).

[85] Jones, D. C., Ruzzo, W. L., Peng, X., and Katze, M. G. Compression of next-generation sequencing reads aided by highly efficient de novo assembly. *Nucleic Acids Research 40*, 22 (Dec. 2012), e171.

[86] Karp, R. M. Reducibility among Combinatorial Problems. In *Complexity of Computer Computations*, R. E. Miller, J. W. Thatcher, and J. D. Bohlinger, Eds., The IBM Research Symposia Series. Springer US, 1972, pp. 85–103.

[87] Kent, W. J. BLAT—The BLAST-Like Alignment Tool. *Genome Research 12*, 4 (2002), 656–664.

[88] Kimura, S., Umeno, M., Skoda, R., Meyer, U., and Gonzalez, F. The human debrisoquine 4-hydroxylase (CYP2D) locus: Sequence and identification of the polymorphic CYP2D6 gene, a related gene, and a pseudogene. *American Journal of Human Genetics 45*, 6 (1989), 889.

[89] Kingsford, C., and Patro, R. Reference-based compression of short-read sequences using path encoding. *Bioinformatics 31*, 12 (June 2015), 1920–1928.

[90] Kirchheiner, J., Schmidt, H., Tzvetkov, M., Keulen, J.-T. H. A., Lötsch, J., Roots, I., and Brockmöller, J. Pharmacokinetics of Codeine and Its Metabolite Morphine in Ultra-Rapid Metabolizers due to CYP2D6 Duplication. *The Pharmacogenomics Journal 7*, 4 (Aug. 2007), 257–265.

[91] Kodama, Y., Shumway, M., and Leinonen, R. The Sequence Read Archive: Explosive Growth of Sequencing Data. *Nucleic Acids Research 40*, Database issue (Jan. 2012), D54–D56.

[92] Korbel, J. O., Urban, A. E., Affourtit, J. P., Godwin, B., Grubert, F., Simons, J. F., Kim, P. M., Palejev, D., Carriero, N. J., Du, L., Taillon, B. E., Chen, Z., Tanzer, A., Saunders, A. C. E., Chi, J., Yang, F., Carter, N. P., Hurles, M. E., Weissman, S. M., Harkins, T. T., Gerstein, M. B., Egholm, M., and Snyder, M. Paired-End Mapping Reveals Extensive Structural Variation in the Human Genome. *Science 318*, 5849 (Oct. 2007), 420–426.

[93] Kozanitis, C., Saunders, C., Kruglyak, S., Bafna, V., and Varghese, G. Compressing Genomic Sequence Fragments Using SlimGene. *Journal of Computational Biology 18*, 3 (2011), 401–413.

[94] Kramer, W. E., Walker, D. L., O'Kane, D. J., Mrazek, D. A., Fisher, P. K., Dukek, B. A., Bruflat, J. K., and Black, J. L. CYP2D6: Novel genomic structures and alleles. *Pharmacogenetics and Genomics 19*, 10 (2009), 813.

[95] Kruskal, J. B. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society 7*, 1 (1956), 48–50.

[96] L. Peter Deutsch. DEFLATE Compressed Data Format Specification version 1.3. https://tools.ietf.org/html/rfc1951.

[97] L. Peter Deutsch. GZIP file format specification version 4.3. https://tools.ietf.org/html/rfc1952.

[98] LaFramboise, T. Single nucleotide polymorphism arrays: A decade of biological, computational and technological advances. *Nucleic Acids Research 37*, 13 (July 2009), 4181–4193.

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

[100] Langmead, B., Trapnell, C., Pop, M., and Salzberg, S. L. Ultrafast and Memory-Efficient Alignment of Short DNA Sequences to the Human Genome. *Genome Biology 10* (2009), R25.

[101] Larkin, M. A., Blackshields, G., Brown, N., Chenna, R., McGettigan, P. A., McWilliam, H., Valentin, F., Wallace, I. M., Wilm, A., Lopez, R., and others. Clustal W and Clustal X version 2.0. *Bioinformatics 23*, 21 (2007), 2947–2948.

[102] Lasse Collin. XZ Utils. http://tukaani.org/xz/.

[103] Lawrence, M. S., Stojanov, P., Mermel, C. H., Garraway, L. A., Golub, T. R., Meyerson, M., Gabriel, S. B., Lander, E. S., and Getz, G. Discovery and Saturation Analysis of Cancer Genes across 21 Tumor Types. *Nature 505*, 7484 (Jan. 2014), 495–501.

[104] Li, H. Difference Between Samtools And Gatk Algorithms. https://www.biostars.org/p/57149/.

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

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

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

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

[109] Liu, L., Li, Y., Li, S., Hu, N., He, Y., Pong, R., Lin, D., Lu, L., Law, M., Liu, L., Li, Y., Li, S., Hu, N., He, Y., Pong, R., Lin, D., Lu, L., and Law, M. Comparison of Next-Generation Sequencing Systems, Comparison of Next-Generation Sequencing Systems. *BioMed Research International 2012, 2012* (July 2012), e251364.

[110] Loeffelholz, M., and Fofanov, Y. The Main Challenges that Remain in Applying High-Throughput Sequencing to Clinical Diagnostics. *Expert Review of Molecular Diagnostics 15*, 11 (2015), 1405–1408.

[111] Londin, E. R., Clark, P., Sponziello, M., Kricka, L. J., Fortina, P., and Park, J. Y. Performance of exome sequencing for pharmacogenomics. *Personalized Medicine 12*, 2 (2014), 109–115.

[112] Ma, Q., and Lu, A. Y. H. Pharmacogenetics, pharmacogenomics, and individualized medicine. *Pharmacological Reviews 63*, 2 (June 2011), 437–459.

[113] Madadi, P., Koren, G., Cairns, J., Chitayat, D., Gaedigk, A., Leeder, J. S., Teitelbaum, R., Karaskov, T., and Aleksa, K. Safety of codeine during breastfeeding. *Canadian Family Physician 53*, 1 (Jan. 2007), 33–35.

[114] Mahoney, M. V. Data Compression Explained. http://mattmahoney.net/dc/dce.html.

[115] Mahoney, M. V. Adaptive weighing of context models for lossless data compression. https://repository.lib.fit.edu/handle/11141/154.

[116] Marco-Sola, S., Sammeth, M., Guigó, R., and Ribeca, P. The GEM Mapper: Fast, Accurate and Versatile Alignment by Filtration. *Nature Methods 9*, 12 (Dec. 2012), 1185–1188.

[117] Mardis, E. R. A Decade/'s Perspective on DNA Sequencing Technology. *Nature 470*, 7333 (Feb. 2011), 198–203.

[118] Margulies, M., Egholm, M., Altman, W. E., Attiya, S., Bader, J. S., Bemben, L. A., Berka, J., Braverman, M. S., Chen, Y.-J., Chen, Z., Dewell, S. B., Du, L., Fierro, J. M., Gomes, X. V., Godwin, B. C., He, W., Helgesen, S., Ho, C. H., Irzyk, G. P., Jando, S. C., Alenquer, M. L. I., Jarvie, T. P., Jirage, K. B., Kim, J.-B., Knight, J. R., Lanza, J. R., Leamon, J. H., Lefkowitz, S. M., Lei, M., Li, J., Lohman, K. L., Lu, H., Makhijani, V. B., McDade, K. E., McKenna, M. P., Myers, E. W., Nickerson, E., Nobile, J. R., Plant, R., Puc, B. P., Ronan, M. T., Roth, G. T., Sarkis, G. J., Simons, J. F., Simpson, J. W., Srinivasan, M., Tartaro, K. R., Tomasz, A., Vogt, K. A., Volkmer, G. A., Wang, S. H., Wang, Y., Weiner, M. P., Yu, P., Begley, R. F., and Rothberg, J. M. Genome Sequencing in Microfabricated High-Density Picolitre Reactors. *Nature 437*, 7057 (Sept. 2005), 376–380.

[119] Massie, M., Nothaft, F., Hartl, C., Kozanitis, C., Schumacher, A., Joseph, A. D., and Patterson, D. A. ADAM: Genomics Formats and Processing Patterns for Cloud Scale Computing. http://www.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-207.html.

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

[121] McKernan, K. J., Peckham, H. E., Costa, G. L., McLaughlin, S. F., Fu, Y., Tsung, E. F., Clouser, C. R., Duncan, C., Ichikawa, J. K., Lee, C. C., Zhang, Z., Ranade, S. S., Dimalanta, E. T., Hyland, F. C., Sokolsky, T. D., Zhang, L., Sheridan, A., Fu, H., Hendrickson, C. L., Li, B., Kotler, L., Stuart, J. R., Malek, J. A., Manning, J. M., Antipova, A. A., Perez, D. S., Moore, M. P., Hayashibara, K. C., Lyons, M. R., Beaudoin, R. E., Coleman, B. E., Laptewicz, M. W., Sannicandro, A. E., Rhodes, M. D., Gottimukkala, R. K., Yang, S., Bafna, V., Bashir, A., MacBride, A., Alkan, C., Kidd, J. M., Eichler, E. E., Reese, M. G., Vega, F. M. D. L., and Blanchard, A. P. Sequence and Structural Variation in a Human Genome Uncovered by Short-Read, Massively Parallel Ligation Sequencing Using Two-Base Encoding. *Genome Research 19*, 9 (Jan. 2009), 1527–1541.

[122] Miclotte, G., Heydari, M., Demeester, P., Rombauts, S., Van de Peer, Y., Audenaert, P., and Fostier, J. Jabba: Hybrid error correction for long sequencing reads. *Algorithms for Molecular Biology 11*, 1 (2016), 1–12.

[123] Mills, R. E., Pittard, W. S., Mullaney, J. M., Farooq, U., Creasy, T. H., Mahurkar, A. A., Kemeza, D. M., Strassler, D. S., Ponting, C. P., Webber, C., and Devine, S. E. Natural Genetic Variation Caused by Small Insertions and Deletions in the Human Genome. *Genome Research 21*, 6 (June 2011), 830–839.

[124] Mullis, K. B. The unusual origin of the polymerase chain reaction. *Scientific American 262*, 4 (Apr. 1990), 56–61, 64–65.

[125] Myers, E. W. Toward Simplifying and Accurately Formulating Fragment Assembly. *Journal of Computational Biology: A Journal of Computational Molecular Cell Biology 2*, 2 (1995), 275–290.

[126] Nagalakshmi, U., Wang, Z., Waern, K., Shou, C., Raha, D., Gerstein, M., and Snyder, M. The Transcriptional Landscape of the Yeast Genome Defined by RNA Sequencing. *Science 320*, 5881 (June 2008), 1344–1349.

[127] National Human Genome Research Institute (NHGRI). The Cost of Sequencing a Human Genome. https://www.genome.gov/27565109/The-Cost-of-Sequencing-a-Human-Genome.

[128] Nelson, M. R., Wegmann, D., Ehm, M. G., Kessner, D., St. Jean, P., Verzilli, C., Shen, J., Tang, Z., Bacanu, S.-A., Fraser, D., Warren, L., Aponte, J., Zawistowski, M., Liu, X., Zhang, H., Zhang, Y., Li, J., Li, Y., Li, L., Woollard, P., Topp, S., Hall, M. D., Nangle, K., Wang, J., Abecasis, G., Cardon, L. R., Zöllner, S., Whittaker, J. C., Chissoe, S. L., Novembre, J., and Mooser, V. An abundance of rare functional variants in 202 drug target genes sequenced in 14,002 people. *Science 337*, 6090 (July 2012), 100–104.

[129] Newton, C. R., Graham, A., Heptinstall, L. E., Powell, S. J., Summers, C., Kalsheker, N., Smith, J. C., and Markham, A. F. Analysis of any point mutation in DNA. The amplification refractory mutation system (ARMS). *Nucleic Acids Research 17*, 7 (Apr. 1989), 2503–2516.

[130] Nicolae, M., Pathak, S., and Rajasekaran, S. LFQC: A lossless compression algorithm for FASTQ files. *Bioinformatics 31*, 20 (Oct. 2015), 3276–3281.

[131] Numanagić, I., Bonfield, J. K., Hach, F., Voges, J., Ostermann, J., Alberti, C., Mattavelli, M., and Sahinalp, S. C. Comparison of high-throughput sequencing data compression tools. *Nature Methods* (Dec. 2016).

[132] Numanagić, I., Malikić, S., Pratt, V. M., Skaar, T. C., Flockhart, D. A., and Sahinalp, S. C. Cypiripi: Exact Genotyping of CYP2D6 Using High-Throughput Sequencing Data. *Bioinformatics 31*, 12 (June 2015), i27–i34.

[133] Ochoa, I., Hernaez, M., and Weissman, T. Aligned genomic data compression via improved modeling. *Journal of Bioinformatics and Computational Biology 12*, 6 (Dec. 2014), 1442002.

[134] Onsongo, G., Erdmann, J., Spears, M. D., Chilton, J., Beckman, K. B., Hauge, A., Yohe, S., Schomaker, M., Bower, M., Silverstein, K. A. T., and Thyagarajan, B. Implementation of Cloud Based Next Generation Sequencing Data Analysis in a Clinical Laboratory. *BMC Research Notes 7* (2014), 314.

[135] Oscarson, M., McLellan, R. A., Asp, V., Ledesma, M., Ruiz, M. L. B., Sinues, B., Rautio, A., and Ingelman-Sundberg, M. Characterization of a novel CYP2A7/CYP2A6 hybrid allele (CYP2A6*12) that causes reduced CYP2A6 activity. *Human Mutation 20*, 4 (Oct. 2002), 275–283.

[136] Panserat, S., Mura, C., Gérard, N., Vincent-Viry, M., Galteau, M. M., Jacoz-Aigrain, E., and Krishnamoorthy, R. An Unequal Cross-over Event within the CYP2D Gene Cluster Generates a Chimeric CYP2D7/CYP2D6 Gene Which Is Associated with the Poor Metabolizer Phenotype. *British Journal of Clinical Pharmacology 40*, 4 (Oct. 1995), 361–367.

[137] Paridaens, T., Panneel, J., De Neve, W., Lambert, P., and Van de Walle, R. Leveraging CABAC for no-reference compression of genomic data with random access support. In *Data Compression Conference (DCC)* (2016), IEEE Signal Processing Society, pp. 625–625.

[138] Patro, R., and Kingsford, C. Data-dependent bucketing improves reference-free compression of sequencing reads. *Bioinformatics 31*, 17 (Sept. 2015), 2770–2777.

[139] Pavlov, I. 7z Format. http://www.7-zip.org/7z.html.

[140] Pearson, W. R., and Lipman, D. J. Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences of the United States of America 85*, 8 (Apr. 1988), 2444–2448.

[141] Pianezza, M. L., Sellers, E. M., and Tyndale, R. F. Nicotine metabolism defect reduces smoking. *Nature 393*, 6687 (June 1998), 750–750.

[142] Pratt, V. M., Everts, R. E., Aggarwal, P., Beyer, B. N., Broeckel, U., Epstein-Baak, R., Hujsak, P., Kornreich, R., Liao, J., Lorier, R., Scott, S. A., Smith, C. H., Toji, L. H., Turner, A., and Kalman, L. V. Characterization of 137 Genomic DNA Reference Materials for 28 Pharmacogenetic Genes: A GeT-RM Collaborative Project. *The Journal of Molecular Diagnostics 18*, 1 (Jan. 2016), 109–123.

[143] Pratt, V. M., Zehnbauer, B., Wilson, J. A., Baak, R., Babic, N., Bettinotti, M., Buller, A., Butz, K., Campbell, M., Civalier, C., and others. Characterization of 107 Genomic DNA Reference Materials for CYP2D6, CYP2C19, CYP2C9, VKORC1, and UGT1A1: A GeT-RM and Association for Molecular Pathology Collaborative Project. *The Journal of Molecular Diagnostics 12*, 6 (2010), 835–846.

[144] Precision Medicine Initiative (PMI) Working Group. The Precision Medicine Initiative Cohort Program – Building a Research Foundation for 21st Century Medicine. Tech. rep.

[145] Pushkarev, D., Neff, N. F., and Quake, S. R. Single-molecule sequencing of an individual human genome. *Nature Biotechnology 27*, 9 (Sept. 2009), 847–850.

[146] Ragoussis, J. Genotyping technologies for all. *Drug Discovery Today: Technologies 3*, 2 (2006), 115–122.

[147] Raimundo, S., Fischer, J., Eichelbaum, M., Griese, E. U., Schwab, M., and Zanger, U. M. Elucidation of the genetic basis of the common 'intermediate metabolizer' phenotype for drug oxidation by CYP2D6. *Pharmacogenetics 10*, 7 (Oct. 2000), 577–581.

[148] Raunio, H., Rautio, A., Gullstén, H., and Pelkonen, O. Polymorphisms of CYP2A6 and its practical consequences. *British Journal of Clinical Pharmacology 52*, 4 (Oct. 2001), 357–363.

[149] Reuter, J. A., Spacek, D. V., and Snyder, M. P. High-Throughput Sequencing Technologies. *Molecular Cell 58*, 4 (May 2015), 586–597.

[150] Rice, R., and Plaunt, J. Adaptive Variable-Length Coding for Efficient Compression of Spacecraft Television Data. *IEEE Transactions on Communication Technology 19*, 6 (Dec. 1971), 889–897.

[151] Riffel, A. K., Dehghani, M., Hartshorne, T., Floyd, K. C., Leeder, J. S., Rosenblatt, K. P., and Gaedigk, A. CYP2D7 Sequence Variation Interferes with TaqMan CYP2D6 (*) 15 and (*) 35 Genotyping. *Frontiers in Pharmacology 6* (2015), 312.

[152] Rissanen, J., and Langdon, Jr., G. Universal Modeling and Coding. *IEEE Trans. Inf. Theor. 27*, 1 (Jan. 1981), 12–23.

[153] Rissanen, J. J. Generalized Kraft Inequality and Arithmetic Coding. *IBM J. Res. Dev. 20*, 3 (May 1976), 198–203.

[154] Robarge, J. D., Li, L., Desta, Z., Nguyen, A., and Flockhart, D. A. The star-allele nomenclature: Retooling for translational genomics. *Clinical Pharmacology and Therapeutics 82*, 3 (Sept. 2007), 244–248.

[155] Robertson, G., Schein, J., Chiu, R., Corbett, R., Field, M., Jackman, S. D., Mungall, K., Lee, S., Okada, H. M., Qian, J. Q., Griffith, M., Raymond, A., Thiessen, N., Cezard, T., Butterfield, Y. S., Newsome, R., Chan, S. K., She, R., Varhol, R., Kamoh, B., Prabhu, A.-L., Tam, A., Zhao, Y., Moore, R. A., Hirst, M., Marra, M. A., Jones, S. J. M., Hoodless, P. A., and Birol, I. De Novo Assembly and Analysis of RNA-Seq Data. *Nature Methods 7*, 11 (Nov. 2010), 909–912.

[156] Robinson, A. H., and Cherry, C. Results of a prototype television bandwidth compression scheme. *Proceedings of the IEEE 55*, 3 (Mar. 1967), 356–364.

[157] Roguski, L., and Deorowicz, S. DSRC 2–Industry-oriented compression of FASTQ files. *Bioinformatics 30*, 15 (Aug. 2014), 2213–2215.

[158] Sanger, F., Nicklen, S., and Coulson, A. R. DNA Sequencing with Chain-Terminating Inhibitors. *Proceedings of the National Academy of Sciences of the United States of America 74*, 12 (Dec. 1977), 5463–5467.

[159] Sayood, K. *Introduction to Data Compression*. Morgan Kaufmann Publishers, San Francisco, 2000.

[160] Schloss, J. A. How to Get Genomes at One Ten-Thousandth the Cost. *Nature Biotechnology 26*, 10 (Oct. 2008), 1113–1115.

[161] Schoedel, K. A., Hoffmann, E. B., Rao, Y., Sellers, E. M., and Tyndale, R. F. Ethnic variation in CYP2A6 and association of genetically slow nicotine metabolism and smoking in adult Caucasians. *Pharmacogenetics 14*, 9 (Sept. 2004), 615–626.

[162] Service, R. F. The Race for the $1000 Genome. *Science 311*, 5767 (Mar. 2006), 1544–1546.

[163] Seward, J. bzip2 and libbzip2. http://www.bzip.org/index.html.

[164] Shannon, C. E., and Weaver, W. *The Mathematical Theory of Communication*. University of Illinois Press, Sept. 1998.

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

[166] Sim, S. C., and Ingelman-Sundberg, M. The Human Cytochrome P450 (CYP) Allele Nomenclature Website: A Peer-Reviewed Database of CYP Variants and their Associated Effects. *Human Genomics 4*, 4 (Apr. 2010), 278–281.

[167] Stephens, Z. D., Lee, S. Y., Faghri, F., Campbell, R. H., Zhai, C., Efron, M. J., Iyer, R., Schatz, M. C., Sinha, S., and Robinson, G. E. Big Data: Astronomical or Genomical? *PLoS Biology 13*, 7 (July 2015), e1002195.

[168] Tange, O. GNU Parallel - The Command-Line Power Tool. *;login: The USENIX Magazine 36*, 1 (Feb. 2011), 42–47.

[169] Tarasov, A., Vilella, A. J., Cuppen, E., Nijman, I. J., and Prins, P. Sambamba: Fast processing of NGS alignment formats. *Bioinformatics 31*, 12 (June 2015), 2032–2034.

[170] Tembe, W., Lowey, J., and Suh, E. G-SQZ: Compact encoding of genomic sequence and quality data. *Bioinformatics 26*, 17 (2010), 2192–2194.

[171] The Cancer Genome Atlas Research Network, Weinstein, J. N., Collisson, E. A., Mills, G. B., Shaw, K. R. M., Ozenberger, B. A., Ellrott, K., Shmulevich, I., Sander, C., and Stuart, J. M. The Cancer Genome Atlas Pan-Cancer Analysis Project. *Nature Genetics 45*, 10 (Oct. 2013), 1113–1120.

[172] The CRAM Format Specification Working Group. CRAM format specification (version 3.0). http://samtools.github.io/hts-specs/CRAMv3.pdf.

[173] The International Warfarin Pharmacogenetics Consortium. Estimation of the Warfarin Dose with Clinical and Pharmacogenetic Data. *New England Journal of Medicine 360*, 8 (Feb. 2009), 753–764.

[174] The SAM/BAM Format Specification Working Group. *Sequence Alignment/Map Format Specification*. http://samtools.github.io/hts-specs/SAMv1.pdf.

[175] Tremaine, L., Brian, W., DelMonte, T., Francke, S., Groenen, P., Johnson, K., Li, L., Pearson, K., and Marshall, J.-C. The Role of ADME Pharmacogenomics in Early Clinical Trials: Perspective of the Industry Pharmacogenomics Working Group (I-PWG). *Pharmacogenomics 16*, 18 (Nov. 2015), 2055–2067.

[176] Twist, G. P., Gaedigk, A., Miller, N. A., Farrow, E. G., Willig, L. K., Dinwiddie, D. L., Petrikin, J. E., Soden, S. E., Herd, S., Gibson, M., Cakici, J. A., Riffel, A. K., Leeder, J. S., Dinakarpandian, D., and Kingsmore, S. F. Constellation: A tool for rapid, automated phenotype assignment of a highly polymorphic pharmacogene, CYP2D6, from whole-genome sequences. *npj Genomic Medicine 1* (Jan. 2016), 15007.

[177] Van der Auwera, G. A., Carneiro, M. O., Hartl, C., Poplin, R., del Angel, G., Levy-Moonshine, A., Jordan, T., Shakir, K., Roazen, D., Thibault, J., Banks, E., Garimella, K. V., Altshuler, D., Gabriel, S., and DePristo, M. A. From FastQ data to high confidence variant calls: The Genome Analysis Toolkit best practices pipeline. *Current Protocols in Bioinformatics 11*, 1110 (Oct. 2013), 11.10.1–11.10.33.

[178] Voges, J., Munderloh, M., and Ostermann, J. Predictive Coding of Aligned Next-Generation Sequencing Data (Accepted for publication). In *Data Compression Conference (DCC)* (2016). Accepted for publication.

[179] Wan, R., Anh, V. N., and Asai, K. Transformations for the compression of FASTQ quality scores of next-generation sequencing data. *Bioinformatics 28*, 5 (2012), 628–635.

[180] Wang, Y., Yang, Q., and Wang, Z. The Evolution of Nanopore Sequencing. *Frontiers in Genetics 5* (2014), 449.

[181] Wassenaar, C. A., Zhou, Q., and Tyndale, R. F. CYP2A6 genotyping methods and strategies using real-time and end point PCR platforms. *Pharmacogenomics 17*, 2 (Dec. 2015), 147–162.

[182] Watson, M. Illuminating the Future of DNA Sequencing. *Genome Biology 15*, 2 (2014), 108.

[183] Welch, T. A. A Technique for High-Performance Data Compression. *Computer 17*, 6 (June 1984), 8–19.

[184] Wikipedia, The Free Encyclopedia. Burrows–Wheeler transform. In *Burrows–Wheeler Transform*. Mar. 2016. https://en.wikipedia.org/w/index.php?title=Burrows%E2%80%93Wheeler_transform&oldid=707982608.

[185] Worthey, E. A., Mayer, A. N., Syverson, G. D., Helbling, D., Bonacci, B. B., Decker, B., Serpe, J. M., Dasu, T., Tschannen, M. R., Veith, R. L., Basehore, M. J., Broeckel, U., Tomita-Mitchell, A., Arca, M. J., Casper, J. T., Margolis, D. A., Bick, D. P., Hessner, M. J., Routes, J. M., Verbsky, J. W., Jacob, H. J., and Dimmock, D. P. Making a Definitive Diagnosis: Successful Clinical Application of Whole Exome Sequencing in a Child with Intractable Inflammatory Bowel Disease. *Genetics in Medicine: Official Journal of the American College of Medical Genetics 13*, 3 (Mar. 2011), 255–262.

[186] Wu, T. D., and Nacu, S. Fast and SNP-tolerant detection of complex variants and splicing in short reads. *Bioinformatics 26*, 7 (Apr. 2010), 873–881.

[187] Yang, Y., Muzny, D. M., Reid, J. G., Bainbridge, M. N., Willis, A., Ward, P. A., Braxton, A., Beuten, J., Xia, F., Niu, Z., Hardison, M., Person, R., Bekheirnia, M. R., Leduc, M. S., Kirby, A., Pham, P., Scull, J., Wang, M., Ding, Y., Plon, S. E., Lupski, J. R., Beaudet, A. L., Gibbs, R. A., and Eng, C. M. Clinical Whole-Exome Sequencing for the Diagnosis of Mendelian Disorders. *The New England Journal of Medicine 369*, 16 (Oct. 2013), 1502–1511.

[188] Yanovsky, V. ReCoil - an Algorithm for Compression of Extremely Large Datasets of DNA Data. *Algorithms Mol Biol 6* (Oct. 2011), 23.

[189] Zhang, Y., Li, L., Yang, Y., Yang, X., He, S., and Zhu, Z. Light-weight reference-based compression of FASTQ data. *BMC Bioinformatics 16*, 1 (June 2015).

[190] Zhang, Y., Patel, K., Endrawis, T., Bowers, A., and Sun, Y. A FASTQ compressor based on integer-mapped k-mer indexing for biologist. *Gene 579*, 1 (Mar. 2016), 75–81.

[191] Zhou, S.-F. Polymorphism of human cytochrome P450 2D6 and its clinical significance. *Clinical Pharmacokinetics 48*, 12 (2009), 761–804.

[192] Ziv, J., and Lempel, A. A Universal Algorithm for Sequential Data Compression. *IEEE Transactions on Information Theory 23*, 3 (1977), 337–343.

[193] Ziv, J., and Lempel, A. Compression of Individual Sequences via Variable-Rate Coding. *IEEE Transactions on Information Theory 24*, 5 (1978), 530–536.

[194] Zook, J. M., Chapman, B., Wang, J., Mittelman, D., Hofmann, O., Hide, W., and Salit, M. Integrating human sequence data sets provides a resource of benchmark SNP and indel genotype calls. *Nature Biotechnology 32* (2014), 246–251.

Appendix A

DeeZ Materials

The following reference genomes were used:

- UCSC *H.Sapiens* hg19 (http://hgdownload.cse.ucsc.edu/goldenPath/hg19/bigZips/chromFa.tar.gz)
- *P.aeruginosa* PAO1 chromosome (http://www.ncbi.nlm.nih.gov/nuccore/110645304)
- *E.Coli* DH10B (https://raw.githubusercontent.com/allanroscoche/PathTree/master/data/DH10B_WithDup_FinalEdit_validated.fasta)

The following parameters were used for the invocation of each tool:

Gzip Compression: `gzip input.sam -c >input.gz`;
Decompression: `gzip -d input.gz -c >input_dc.sam`

SAMtools v0.1.19 Compression: `samtools view -bS input.sam >input.bam`;
Decompression: `samtools view -h input.bam >input.sam`

Cramtools v2.0 Compression: `java -Xmx8g -jar cramtools-2.0.jar cram -I input.sam -capture-all-tags -input-is-sam -Q -n -R reference.fa >input.cram`;
Decompression: `java -Xmx8g -jar cramtools-2.0.jar bam -I input.cram -print-sam-header -R reference.fa >input_dc.sam`

Scramble v1.13.7 Compression: `scramble -I sam -O cram -r reference.fa input.sam >input.scr`;
Decompression: `scramble -I cram -O sam -r reference.fa input.scr >input_dc.sam`

Quip v1.1.6 Compression: `quip input.sam -c >input.qp`;
Decompression: `quip input.qp -output=sam -d -c >input_dc.sam`.
Reference-based invocation included `-r reference.fa` parameter.

sam_comp v0.7 and v0.8 Compression: `sam_comp <input.sam >input.zam`;
Decompression: `sam_comp -d <input.zam >input_dc.sam`.
Reference-based invocation included `sam_comp -r reference`.

Goby v2.3.4 Compression: `java -Xmx8g -jar goby.jar -m stc -preserve-all-mapped-qualities -preserve-all-tags -preserve-soft-clips -preserve-read-names -x AlignmentWriterImpl:permutate-query-indices=false -x SAMToCompactMode:ignore-read-origin=false -x MessageChunksWriter:codec=hybrid-1 -x AlignmentCollectionHandler:enable-domain-optimizations=true -x MessageChunksWriter:compressing-codec=true -g reference.fa -i input.sam -o input.goby`;
Decompression: `java -Xmx8g -jar goby.jar -m cts -g reference.fa input.goby -o input_dc.sam.sam`

DeeZ Compression: `deez -r reference.fa input.sam -c >input.dz`;
Decompression: `deez -r reference.fa input.dz -c >input_dc.sam`.
sam_comp mode included `-q1` parameter.
Lossy quality mode included `-l30` parameter.

Appendix B

Compression Benchmarking Materials

Table B.1: A summary of evaluated tools and their versions.

Tool	Version	URL
pigz	2.3.3	http://zlib.net/pigz/pigz-2.3.3.tar.gz
bzip2	1.0.6	http://www.bzip.org/1.0.6/bzip2-1.0.6.tar.gz
pbzip2	1.1.12	https://launchpad.net/pbzip2/1.1/1.1.12/+download/pbzip2-1.1.12.tar.gz
Cramtools	0c89dd8	https://github.com/enasequence/cramtools.git
Scramble	1.14.6	http://sourceforge.net/projects/staden/files/io_lib/1.14.0/io_lib-1.14.0.tar.gz/download
Sambamba	0.6.0	https://github.com/lomereiter/sambamba/releases/download/v0.6.0/sambamba_v0.6.0_linux.tar.bz2
Samtools	e2bb18f	https://github.com/samtools/samtools
HTS Lib	897a34f	https://github.com/samtools/htslib
sam_comp	0.7	http://sourceforge.net/projects/samcomp/files/latest/download
TSC	1.5	N/A
CBC	d18afbb	https://github.com/mikelhernaez/cbc
Picard	1.138	https://github.com/broadinstitute/picard/releases/download/1.138/picard-tools-1.138.zip
Quip	629f6fc	https://github.com/dcjones/quip
DeeZ	abc6dc8 (v1.9 beta 1)	https://github.com/sfu-compbio/deez
Fqzcomp	4.6	http://sourceforge.net/projects/fqzcomp/files/latest/download
DSRC2	5eda82c	https://github.com/lrog/dsrc
Fastqz	15 / 39b2bbc	https://github.com/fwip/fastqz
Slimfastq	f55ae88	https://github.com/Infinidat/slimfastq
BEETL	6c240ea	https://github.com/BEETL/BEETL
ORCOM	6280813	https://github.com/lrog/orcom
LFQC	1.1 (b6bc1b8)	https://github.com/mariusmni/lfqc
k-Path	0.6.3	http://www.cs.cmu.edu/~ckingsf/software/pathenc/kpath-0.6.3.tar.gz
Mince	3ddc3a1	https://github.com/Kingsford-Group/mince
LW-FQZip	1.02	http://csse.szu.edu.cn/staff/zhuzx/LWFQZip/LWFQZip-v1.02.zip
FQC	3.0c	http://metagenomics.atc.tcs.com/Compression_archive/FQC/FQC_LINUX_64bit.tar.gz
Leon	1.0.0	http://gatb-tools.gforge.inria.fr/versions/src/leon-1.0.0-Source.tar.gz
SCALCE	ef1bc4c (v2.8)	https://github.com/sfu-compbio/scalce.git
SRA	2.5.2	http://ftp-trace.ncbi.nlm.nih.gov/sra/sdk/2.5.2/sratoolkit.2.5.2-centos_linux64.tar.gz

Table B.2: A summary of crashes and incorrect output observed in some tools. Dark green indicates perfect decompression. Light green indicates cosmetic changes in the output file (e.g. lack of comments, different optional field order etc.). Orange indicates slightly higher level of corruption, with a small impact to the original file (e.g. slightly modified paired-end information, different read identifiers etc.). Magenta indicates either crash or completely corrupted output files.

Sample	SRR554369	SRR327342	MH0001.081C	SRR1284073	SRR870667	ERR174310	ERP001775
DSRC2				Does not support variable read lengths			Not tested
Fqzcomp	Does not support comment field						
	Crashes during decompression	Invalid N qualities	Crashes during decompression	Does not support variable read lengths	Crashes during decompression	Invalid N qualities	Not tested
Fastqz	Crashes during compression: cannot read FASTQ files with comments					Invalid output	Not tested
Slimfastq				Does not support variable read lengths			
FQC							Not tested
LFQC	Does not support FASTQ comments						
				Invalid output	Decompression too slow	Compression too slow	Not tested
SCALCE	Does not support comment field and read identifier comments Invalid N qualities						
				Invalid output			
LW-FQZip	Slight read identifier and comment field corruption	Slight read identifier and comment field corruption	Crashes during compression	Does not support variable read lengths	Slight read identifier and comment field corruption	Crashes during compression	Not tested
Quip	Does not support FASTQ comments						
Leon	Does not support FASTQ comments						
				Slightly corrupted output			
KIC							Not tested
Orcom				Does not support variable read lengths			
BEETL				Does not support variable read lengths			Not tested
k-Path				Does not support variable read lengths			Not tested
Mince		Does not support paired-end library with different read sizes	Wrong result	Does not support variable read lengths			Not tested

(a) FASTQ tools

Sample	DH10B	9827.2.49	sample-2-1	K562.LID8	dm3	NA12878.P	HCC1954	NA12878.S1
Scramble	RG, MD and NM tags are changed or ignored in the reference mode (with default parameters) SAM comment is slightly modified							Invalid qualities and CIGARs are ignored
Picard	SAM comment is slightly modified							Not tested
Sambamba	SAM comment is slightly modified							
CBC	Crashes during compression (authors working on the fix)							
Quip	Slight, non-functional modifications to the paired-end information Order of optional fields is changed							
		Reference mode cannot handle invalid mappings	Reference mode cannot handle invalid mappings			Reference mode cannot handle invalid mappings	Reference mode cannot handle invalid mappings	
Cramtools	RG, MD and NM tags are changed or ignored in the reference mode SAM comment is slightly modified Paired-end information is modified							Not tested
sam_comp	Paired-end information and optional fields are ignored							

(b) SAM tools

Appendix C

Cypiripi Materials

The NA12878 assembly was accessed from http://sv.gersteinlab.org/NA12878_diploid/. The Illumina HiSeq runfile for simNGS was obtained from http://www.ebi.ac.uk/goldman-srv/simNGS/runfiles/101cycleHiSeq/s_3_4x.runfile.

simNGS was invoked as:

```
simLibrary -x [coverage] [fasta] | simNGS -o fastq -p paired [runfile]
```

Cypiripi was invoked as:

```
cypiripi -C [coverage] -T [eta] -r [library] -s [mapping.sam]
```

Fusion datasets were run with the addition of `-F` parameter.