# Large-Scale Analysis of Population Structural Variants Using Terabytes of SRA Sequencing Data

By

MOUSTAFA SHOKROF
DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

In

Computer Science

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

C. Titus Brown, Chair

Megan Dennis

Fereydoun Hormozdiari

Committee in Charge

2024

*To Reem, No words can adequately summarize your role in my PhD journey, in bringing this dissertation to light, and more importantly, in shaping the man I am today. Although I haven't walked on water yet, I will never forget your love and support ...*
*To Mom and Dad, No matter how physically close or far, starting from living in Mom's womb to sleeping in the same room, then a different room, then a different city, and then a different continent, you always had my back. I couldn't have achieved anything without your support and unconditional love ...*
*To Lobna, Reem, and Mona, who complete me: each of you helps me in her unique way, and each of you holds a unique place in my heart*

# Contents

# List of Figures

# List of Tables

## Abstract

**Large-Scale Analysis of Population Structural Variants Using Terabytes of SRA Sequencing Data**

Studying structural variants (SV) in populations is crucial since they cause more diversity and have more effects on gene function than small variants. However, population scale studies are challenging since finding SV from inexpensive short-read sequencing(SRS) methods have a high false positive rate. Conversely, long-read sequencing(LRS) are more accurate for SV discovery but are expensive at a population scale. Here, I develop new unbiased techniques to study SV in populations that are more scalable than the state-of-the-art. I show their utility in creating an SV catalog for the cattle breed augmented with their allele frequency.

## Acknowledgments

To Tamer, who believed in me and helped me get back to research when I always lost hope. I will always be grateful for teaching me critical thinking and conducting research. I am grateful for all life advice. Even if our paths may diverge, you will always be a big brother to me.

To Titus, who showed me what a great boss truly is. Thanks for providing a safe, financially secure environment for me to grow and prosper in my creativity. Also, thanks for all your guidance and advice; I wouldn't have been able to submit this thesis without you.

To Dib-lab members, I have never witnessed anything from you other than kindness, support, and generosity. To Mo, who is very resourceful and always there to help at any time. To Colton, who has helped me many times in revising my presentations and CV, providing well-thought-out feedback.

To Lucas, who had my back many times during the PhD journey. Knowing him as a friend is one of the best things that happened to me in the last 5 years. I am glad and thankful to have you in my life. Also, thanks for introducing Michi to me.

To my Mexican brother, Javier, you treated me as part of your family, and that's how I will always treat you.

To Melissa and Wennie, thanks for all your care no matter how far we are. Thanks for providing warmth when the world is so cold.

To Gehan and Mahmoud, who treated me as their own son all the time, even in the toughest moments of my life. I will never forget that, and I will be grateful forever.

# Chapter 1

# Introduction

Genomic data has experienced explosive growth in the last decade [1], and identifying genomic variation among these genomes is crucial for understanding how genomes shape us and all living organisms. This understanding can aid in curing diseases [2], advancing personalized medicine [3], and enhancing the breeding of plants [4] and animals [5]. Genomic variation can fall into two categories: small variants, including single nucleotide variations and insertions/deletions of a size less than 50 bp; and structural variants, variations of a size greater than 50 bp. SVs can be insertions, deletions, duplications, inversions, copy number variations, or even a complex composite of different rearrangements. Here, we are focusing on studying structural variations between genomes in populations.

Studying SVs in populations presents several challenges. First, sequencing a large number of genomes using Short Read Sequencing (SRS) is affordable, but discovering SV from SRS samples has a low recall [6]. On the other hand, Long Read Sequencing (LRS) is more accurate and comprehensive but expensive for population-scale studies. Moreover, population studies require analyzing a huge number of samples, which poses both challenges and opportunities. Scalable tools are required for transferring, storing, and analyzing terabytes of data. However, analyzing a large number of samples opens the opportunity to leverage population information to increase the accuracy of individual analyses. Therefore, programs that can take advantage of the abundance of SRS samples in a joint analysis of SVs in the population will represent a significant advancement to the field.

The traditional approach to population-scale SV studies relies on aligning each sequence from SRS or LRS to a linear genome [7, 8]. This process generates BAM files. SVs are identified separately in each sample's BAM file [9, 10, 11, 12, 13]. Later, these SV call sets are combined to create SV catalogs and to calculate allele frequencies [14]. We can enhance the catalog by re-genotyping the discovered variants in the SRS mapping files [15, 16]. Aligning to linear genomes allows reads to be sorted by position. This enables the grouping of reads from the same regions for the analysis of variations, including small and structural variants. Also, it facilitates the aggregation of individual variant call sets for population-level analysis. This is done by providing a coordinate system to compare variants from different samples.

However, this method introduces biases associated with the reference genomes. Many studies have pointed out these biases in variant analysis [17, 18, 19]. The biases are especially evident when a single ethnicity is predominant in the reference genome, such as the European ethnicity in GRCh38. The issue is even more significant in non-human genomes. For example, a single linear genome reference may not reflect the full genomic diversity of a species. The assembly of *Bos indicus* genomes [20] revealed 148.5MB of novel sequences not found in the *Bos taurus* ARS-UCD1.2 reference genome [21].

Furthermore, storing sequences in BAM files limits their accessibility for population studies. These files are indexed by their mapping positions to the linear genome. This indexing favors sequences that map perfectly to the reference. However, searching for sequences that do not exist in the genome is problematic. It requires sifting through numerous unmapped reads. It is also challenging to find sequences divergent from the reference genome. This is because the reads containing matches may be misplaced.

Moreover, storing sequence data for population studies often necessitates using cloud storage due to the sheer size of the datasets, which can be problematic. Accessing these large quantities of data from the cloud can be inefficient and costly, posing a significant challenge for researchers conducting population-level studies.

Alternatively, kmer-based methods offer a solution that circumvents the need to align reads to reference genomes. These methods break down reads into fixed-size strings (kmers) and compile a list of unique kmers, noting the frequency of each kmer in the reads. These kmers can be organized into a De Bruijn graph, where each node represents a kmer and edges connect nodes that share a sequence overlap of k-1. Tools like Kevlar [22] utilize kmers to identify reads that contain small variants and SVs for variant calling. Corticall, for example, represents a population of bacteria with a colored De Bruijn graph, distinguishing samples by color to detect de novo mutations. Beyond SV calling, applications such as Pangenie [23] and Nebula [24] genotype SVs by identifying kmers unique to reference and alternative alleles, determining genotypes by the variation in kmer counts within SRS data. In conclusion, kmer comparison is an efficient alternative to alignment-based SV analysis methods however it has some limitations. Repeated and low complexity regions are often collapsed in the De Bruijn graph, making it harder to study variants in those regions.

Moreover, kmers provide a promising solution to the accessibility issues in population studies. Recent advancements in data structures and algorithms have made it possible to store Colored De Bruijn Graphs (CDGs) compactly, while still enabling efficient sequence search capabilities. For example, Fulgor [25] can represent 150,000 Salmonella genomes within a 70GB CDG. This kmer index allows users to query by kmers and receive a list of samples containing that specific kmer, facilitating sequence search functions.

Tools like Metagraph and Reindeer have developed Counting Colored De Bruijn Graphs (CCDGs), which not only return the presence of a kmer but also its count across the indexed samples. Metagraph created a CCDG that encapsulates 2.5K RNA-seq human samples in just 27.6GB. Thus, the kmer approach enables the representation of sequences from thousands of samples in highly efficient data structures, preserving the functionality for sequence searches.

In this thesis, I investigate the use of kmers and their indexing to address challenges in population-scale SV studies. I have developed a data structure known as the MQF (Minimum Quotient Filter) designed to efficiently store De Bruijn graphs with additional metadata, which facilitates the implementation of colored De Bruijn graphs.

Furthermore, using the Metagraph tool, I have constructed two Counting Colored

De Bruijn Graphs (CCDGs) to represent human and cattle populations. The human population CCDG was generated from 4.2K whole-genome sequencing (WGS) samples drawn from 140 populations, condensing 95TB of CRAM files into a 925.2GB CCDG. For the cattle population, I created a 293.8GB CCDG from 1.1K WGS samples spanning 80 breeds. These CCDGs serve as the foundation for the population-scale SV studies presented in this thesis.

Additionally, I have developed a population genotyping workflow named "The Great Genotyper," tailored for genotyping small and structural variants within a population of samples. This workflow has demonstrated exceptional speed and unprecedented scalability in genotyping a large number of samples. This scalability allows the algorithm to leverage population information to enhance genotyping accuracy. Applications of The Great Genotyper include accurate allele frequency estimation, construction of SV imputation panels, and refinement of GWAS SNP mappings using SVs from the human pangenome.

Finally, I developed a workflow to create a catalog for common SV in cattle breeds. The workflow merges the mapping-based SV calling approach with the kmer based population genotyping method implemented in The Great Genotyper. I developed a lenient SV calling workflow that aggregates SVs from all available data while tolerating some false positives. Subsequently, we utilized the efficiency of The Great Genotyper to genotype the identified SVs, compute their allele frequencies, and filter out rare variants, which are likely errors or novel SVs. The resulting catalog surpasses previous ones in breed coverage and the number of discovered variants.

### 1.0.1 Thesis Structure

1. **Chapter 2** introduces the Mixed-Counters Quotient Filter (MQF), an associative data structure that pairs kmers with their counts and additional metadata. It also describes a buffered MQF variant with a quick insertion algorithm, utilizing external memory for most of the data structure.

2. **Chapter 3** presents The Great Genotyper, a workflow for genotyping variants in

population studies using CCDGs rather than raw sequences or mapping files. This chapter demonstrates The Great Genotyper's speed, matching state-of-the-art precision and recall, and explores its potential to transform population genomics.

3. **Chapter 4** details our efforts in developing a common SV catalog for *Bos taurus* breeds. We benchmark various SV calling methods on *Bos taurus* genomes and outline the process of constructing CCDGs for 1.1K cattle samples. Additionally, we propose a novel approach for generating SV catalogs, leveraging population genotyping to remove false positives and rare variants.

# Chapter 2

# MQF

## 2.1 Introduction

Online algorithms effectively support streaming analysis of large data sets, which is important for analyzing data sets with large volume and high velocity[26]. Approximate data structures are commonly used in online algorithms to provide better average space and time efficiency[27]. For example, the Bloom filter supports approximate set membership queries with a predefined false positive rate (FPR)[28]. The count-min sketch (CMS) is similar to Bloom filters and can be used to count items with a tunable rate of overestimation. However, there are a number of problems with Bloom filters and the CMS - in particular, they do not support data locality.

The Counting Quotient Filter (CQF) is a more efficient data structure that serves similar purposes with better efficiency for skewed distributions and much better data locality[29]. The CQF is a recent variant of quotient filters that tracks the count of its items using a variable size counter. As a compact hashtable, CQF can perform in either probabilistic or exact modes and supports deletes, merges, and resizing.

Analysis of k-mers in biological sequencing data sets is an ongoing challenge[30]. K-mers in raw sequencing data often have a Zipfian distribution, and the CQF was built to minimize memory requirements for counting such items[29]. However, this advantage deteriorates in applications that require frequent random access to the data structure, and where the k-mer count distribution may change in response to different sampling approaches, library preparation and/or sequencing technologies. For example, k-mer frequency across 1000s of RNAseq experiments shows different patterns of abundant k-mers[31].

Data structures like CMS[32] and CQF[29] also do not natively support associating k-mers with multiple values, which can be useful for coloring in De Bruijn graphs as well as other features[33, 34, 35]. Classical hash tables are designed to associate their keys with a generic data type but they are expensive memory-wise[36]. Tools like jellyfish[37] and CHTKC[38] use lock free hash tables in its k-mer counting algorithms. Minimal Perfect Hash Functions (MPHFs) can provide a more compact solution by mapping each k-mer into a unique integer. These integers can then be used as indices for the k-mers to label them in other data structures[39]. An implementation capable of handling large scale datasets with fast performance requires  3 bits per element[40]. However, such a concise representation comes with a high false-positive rate on queries for non-existent items. Moreover, unlike hashtables, MPHF does not support insertions or deletions thus any change in the k-mer set would require rehashing of the original dataset.

In this paper, we introduce the mixed-counters quotient filter (MQF), a modified version of the CQF with a new encoding scheme and labeling system supporting high data locality. We further show how Buffered MQF can be used to scale MQF to solid-state disks. We compare between MQF and the CQF, CMS, and MPHF data structures regarding memory efficiency, speed performance, and applicability to specific data analysis challenges. We further do a direct comparison of the CMS to MQF in the khmer software package for sequencing data analysis, to showcase the benefits of MQF is in real world applications.

## 2.2 Results

### 2.2.1 MQF has a lower load factor than CQF

The load factor is defined as the actual space utilized divided by the total space assigned for the data structure, and is an important measure of data structure performance. To compare load factors between the CQF and MQF data structures, instances of both

| Distribution | CQF | MQF |
|---|---|---|
| Z2 | 402 | 671 |
| Z3 | 161 | 161 |
| Z5 | 134 | 134 |
| Kmers | 402 | 671 |
| Uniform | 402 | 1275 |

Table 2.1. Number of items (in millions) inserted to achieve a 90% load factor in compact hash tables.

structures were created using the same number of slots ($2^{27}$). Chunks of items from thirteen datasets with different distributions of item frequencies were inserted iteratively to be counted in both data-structures while recording the load factor after the insertion of each chunk. After the insertion of each chunk, both data structures were checked to confirm having the exact same hashes. The experiments stopped when MQF's load factor reached 90%. MQF had lower loading factors for all tested datasets but the difference was minimal for the dataset with the highest Zipfian distribution (Z=5). The lower the tested Zipfian distribution the lower the loading factor of MQF (Figure 2.1, Figure 2.2). A lower loading factor enabled MQF to accommodate >30% of the CQF capacity from a dataset of real k-mers and to exceed the double CQF capacity with uniform distribution (Figure 2.1 and table 2.1).

### 2.2.2 MQF is usually more memory efficient than CQF

Progressively increasing numbers of items were sampled from the real and Zipfian-simulated datasets. The smallest CQF and MQF to store the same number of items from each dataset were created. To do that, the q parameter of CQF versus the q and $F_{size}$ parameters of MQF were calculated empirically. Starting with a small q, CQF and MQF were tested to count the items. The q was incremented to find the smallest value that enables the structure to hold the items. MQF was more memory efficient for most real k-mers and Zipfian-simulated distributions (Figure 2.3-2.4). The tuning of the $F_{size}$ parameter enables MQF to grow in size gradually compared to CQF which has to double in size to fit the min-

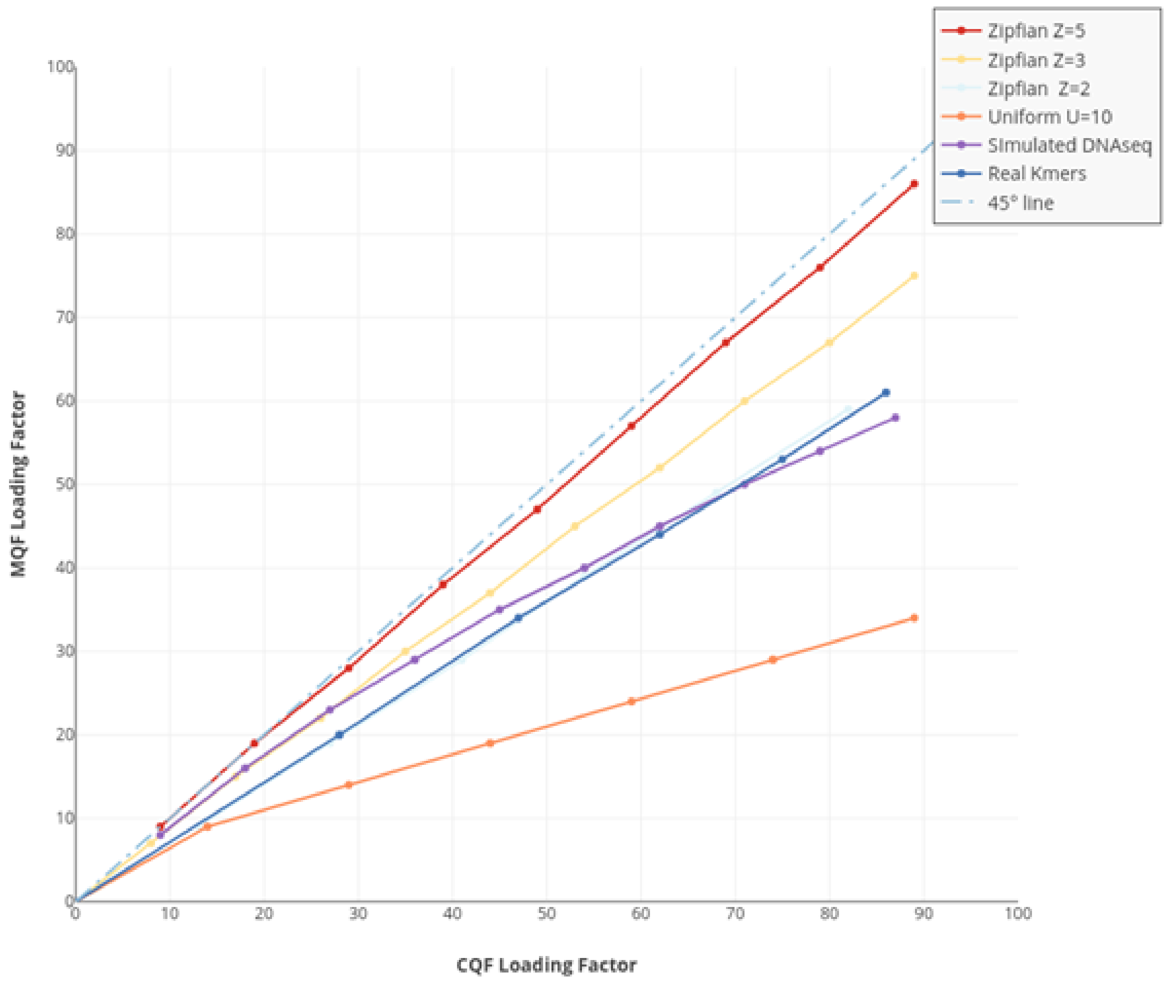

Figure 2.1. MQF has a lower load factor compared to CQF. Chunks of items, from different distributions of item's frequencies, were inserted iteratively to matching CQF and MQF structures. MQF had lower loading factors for all tested datasets with better performance with more uniform distributions (The further from the 45° line the better the MQF).

imal increase in items beyond the capacity of a given q value (Figure 2.5). Therefore, comparing the optimum parameters used to generate both data structures shows that MQF is always more memory efficient if $F_{size} > 1$, while CQF could be slightly more memory efficient if $F_{size} = 1$. This is more likely to happen if sequencing datasets are highly erroneous or very polymorphic (e.g. mitochondrial genome sequencing in SRR12989394). This can also happen if sequencing has a very low coverage, causing most of the k-mers to be single-

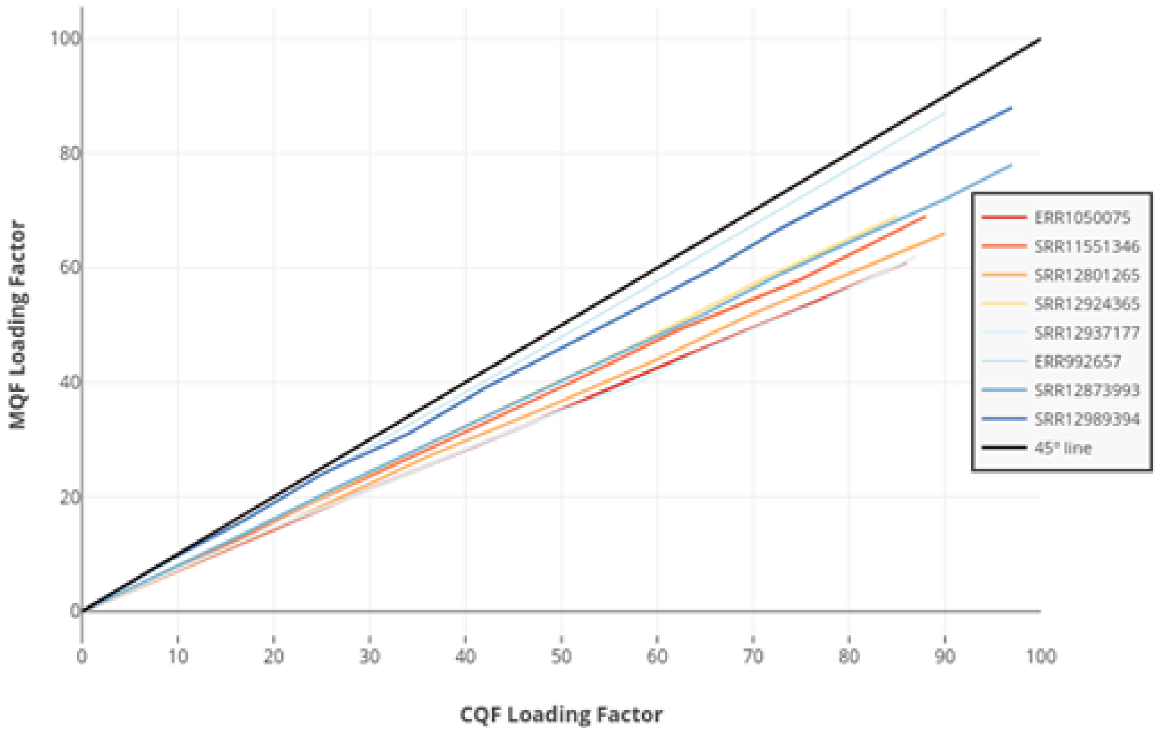

Figure 2.2. MQF has a lower load factor compared to CQF in real datasets. Chunks of items, from 8 real datasets, were inserted iteratively into matching CQF and MQF structures. MQF had lower loading factors for all tested datasets (The further from the 45° line the better the MQF).

tons (e.g. smalls subsets of whole genome sequencing data in ERR992657). MQF comes with a utility script (https://github.com/dib-lab/MQF/blob/master/include/utils.h) to allow prediction of the optimal MQF parameters.

### 2.2.3 MQF is faster than CQF and low-FPR CMS

The in-memory and buffered MQFs were evaluated for speed of insertion and query in comparison to three in-memory counting structures: CQF[29], the original CMS[32], and khmer's CMS[41]. To test the effect of FPR on the performance, the experiment was repeated for 4 different FPRs (0.1, 0.01, 0.001, 0.0001). All tested structures were constructed to have approximately the same memory space except for buffered MQF which used only one-third of this memory for buffering while the full-size filter is on the disk.

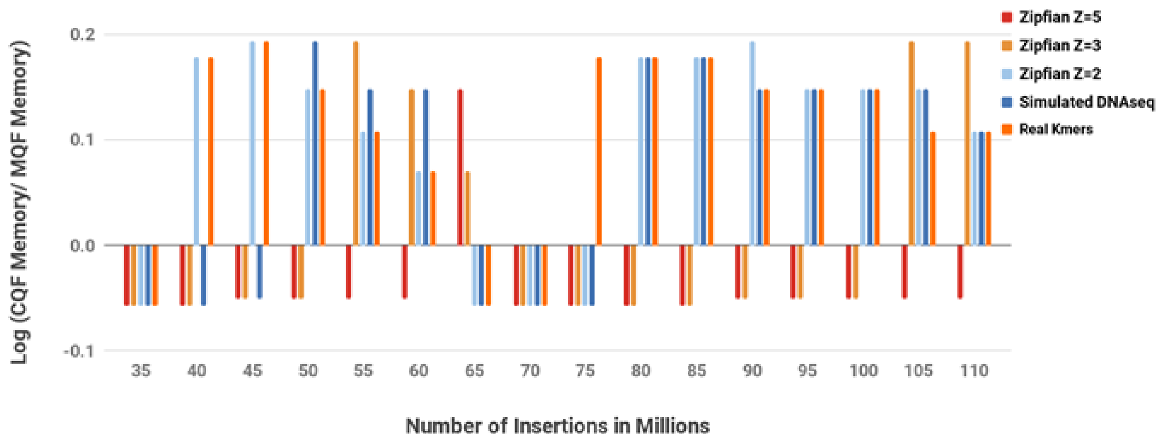

Figure 2.3. **Memory consumption comparison between CQF and MQF.** The graph compares the memory consumption of the smallest CQF and MQF that fits different datasets. The bigger the value on the y-axes, the more memory the MQF saved. Sequencing from ERR1050075 was used to generate a representative dataset for real k-mers.

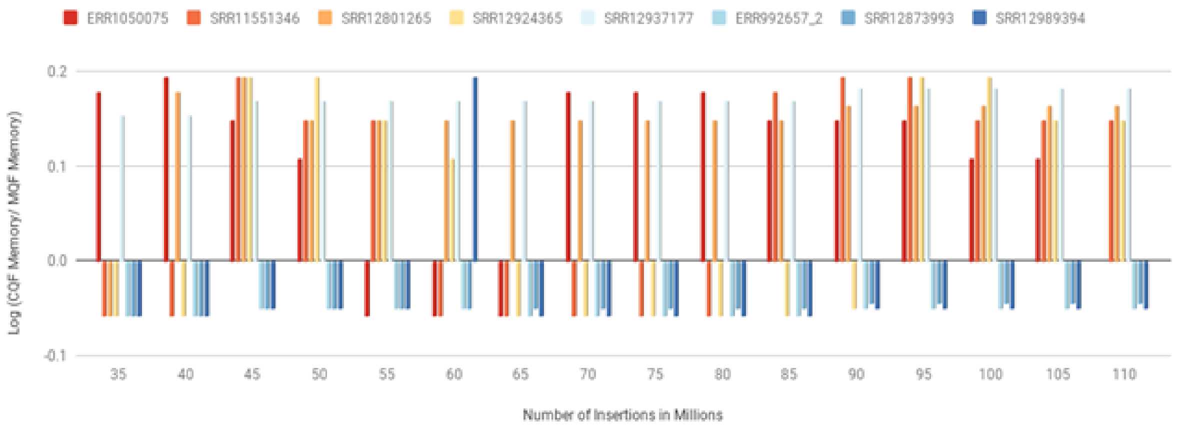

Figure 2.4. **Memory consumption comparison between CQF and MQF in real datasets.** The graph compares the memory consumption of the smallest CQF and MQF that fit different datasets. The bigger the value on the y-axes, the more memory the MQF saved.

MQF is guaranteed to hold the same number of items as a CQF having the same number of slots. The number of slots in CQF was chosen so that the load factor was more than 85% and the MQFs were created with an equal number of slots. Items were sampled to be counted from the real and Zipfian-simulated datasets. After finishing the insertion, to assess the query rate, 5M items from the same distribution as the insertion datasets were queried. Half of the query items didn't exist in the insertion datasets.

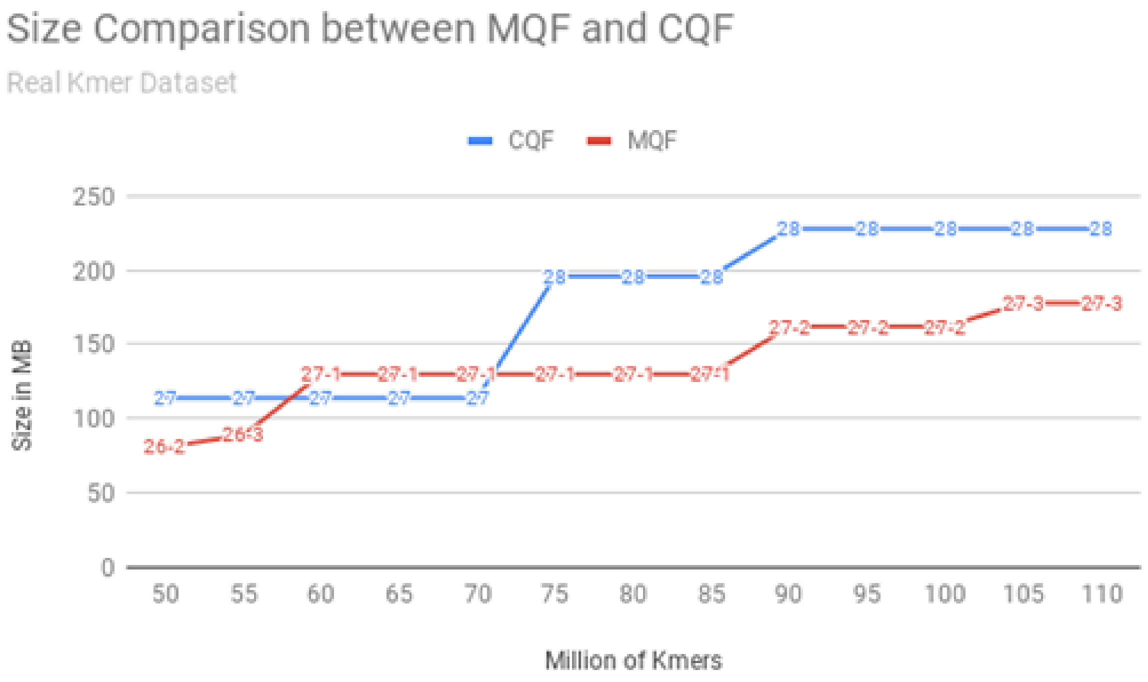

Figure 2.5. **Detailed Memory Consumption Comparison.** The numbers on the CQF curve are the used Q sizes. While the numbers on MQF are (Q size-fixed counter size).

| Distribution | CQF | MQF |
|---|---|---|
| Z2 | 402 | 671 |
| Z3 | 161 | 161 |
| Z5 | 134 | 134 |
| Kmers | 402 | 671 |
| Uniform | 402 | 1275 |

Table 2.2. Number of items (in millions) inserted to achieve a 90% load factor in compact hash tables.

MQF has slightly yet persistent faster insertion and query rates compared to CQF with minimal, if any, effect of the FPR on either structure. The performance of CMS is better with higher FPR and Khmer's implementation of CMS doubles the query rate of the original one. However, MQF is always faster than both CMS unless the FPR is more than 0.01 (Figure 2.6-2.7-2.8).

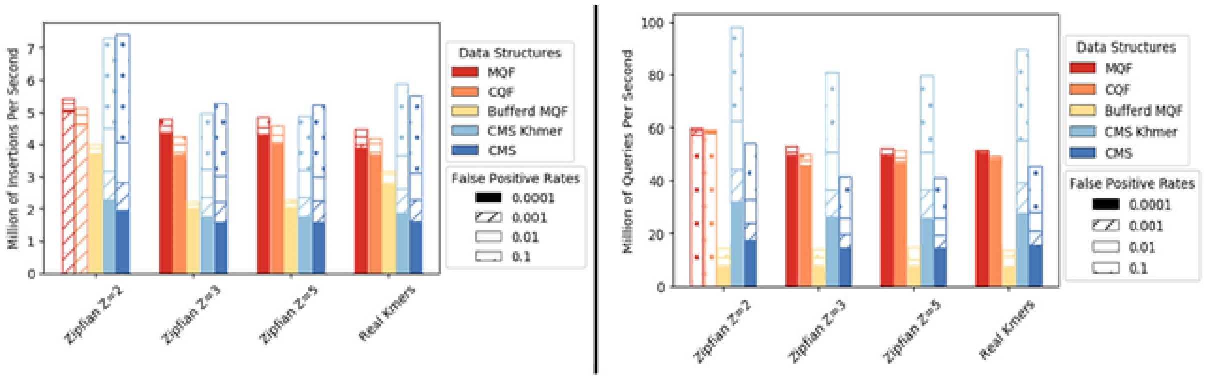

Figure 2.6. **Performance comparison of four data-structures:** MQF, CQF, buffered MQF (using 1/3 the size of other structures), Khmer implementation of CMS, and original implementation of CMS: Insertions rate (left panel) and query rate (right panel). ERR1050075 is used as a representative dataset for real k-mers. Supplementary figure 6 and 7 show the comparisons between 8 real datasets. The performance is plotted as a bar graph where the pattern of each bar shows the cumulative increase in insertions or queries for different false positive rates.

### 2.2.4 MQF outperforms CMS in real-world problems

Khmer is a software package deploying a new implementation of CMS for k-mer counting, error trimming and digital normalization[41]. To test MQF in real-life applications, we assessed the performance of the Khmer software package using CMS versus our new implementation using MQF (https://github.com/dib-lab/khmer/tree/MQFIntegration2). A real RNA seq dataset with 51 million reads from the Genome in a Bottle project[42] was used for error trimming and digital normalization; two real-world applications that involve both k-mer insertions and queries. An exact MQF was used to create a benchmark for the approximate data structures. It took 5Gb RAM to create the data structure and 45 and 43 minutes to perform trimming and digital normalization respectively. The optimal memory for MQF and the optimal number of hash functions for CMS were calculated to achieve the specified false-positive rates. The CMS was constructed with the same size as the corresponding MQFs. The CMS and MQF versions of Khmer were compared regarding the speed and accuracy (Table 2.3).

### 2.2.5 MQF is faster than MPHF

MPHF is constructed by default to fit the input k-mers while MQF would have different load factor that might affect its performance. To address this question, four growing

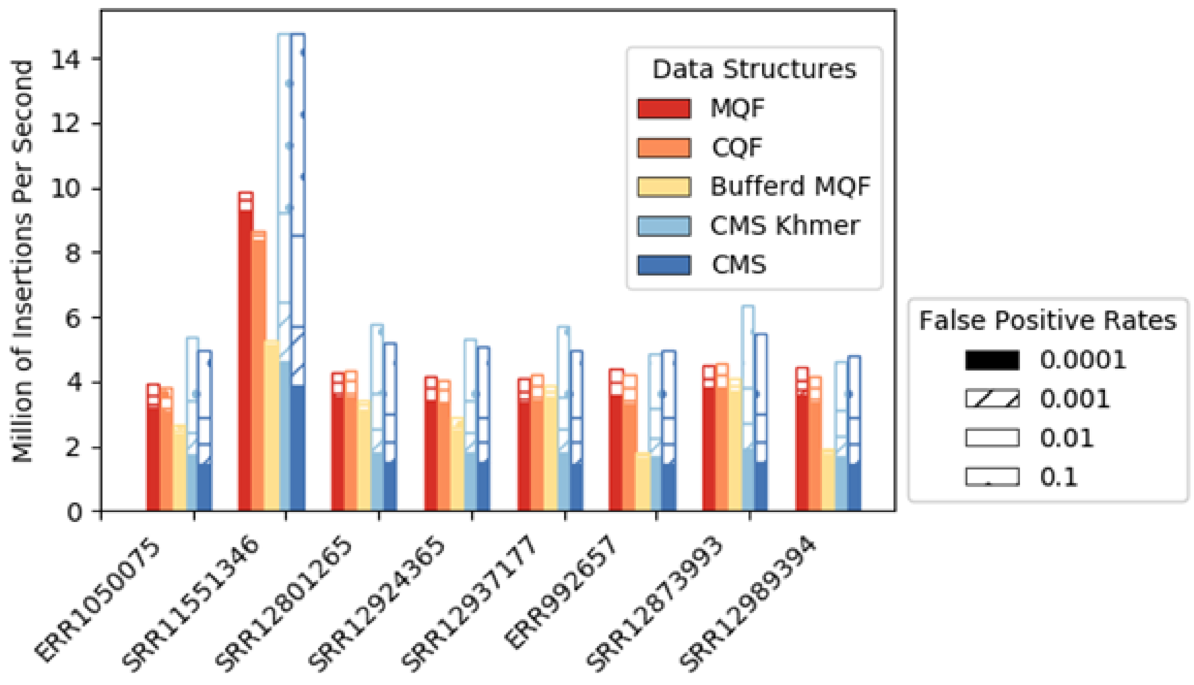

Figure 2.7. **Insertions rate comparison of four data-structures in real datasets:** MQF, CQF, buffered MQF (using 1/3 the size of other structures), Khmer implementation of CMS, and original implementation of CMS. The performance is plotted as a bar graph where the pattern of each bar shows the cumulative increase in insertions for different false positive rates.

| FPR | Memory in GB | Error Trimming | | | | Digital normalization | | | | Error Bound in CMS | Hash func. In CMS |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Time in Min. | | Missed reads with Errors | | Time in Min. | | Reads kept by Error | | | |
| | | MQF | CMS | MQF | CMS | MQF | CMS | MQF | CMS | | |
| 10-1 | 1.8 | 42 | 39 | 11011 | 445817* | 39 | 37 | 3253 | 31143* | 13,11 | 3 |
| 10-2 | 2.6 | 43 | 48 | 1304 | 404354 | 41 | 45 | 416 | 24987 | 14 | 5 |
| 10-3 | 3.4 | 44 | 61 | 130 | 311464 | 42 | 54 | 58 | 21000 | 15 | 7 |
| 10-4 | 4.5 | 44 | 75 | 3 | 292746 | 42 | 68 | 4 | 18449 | 16 | 10 |
| Exact | 5 | 45 | - | 0 | - | 43 | - | 0 | - | - | - |

Table 2.3. **Khmer performance in error trimming and digital normalization using MQF and CMS**. *Percentages of wrong decisions made by CMS at FPR = 0.1 in error trimming and digital normalization are 0.8% and 0.13% of the total number of decisions versus 0.02% and 0.01% made by MQF.

subsets of real k-mers were inserted into MQFs of size 255 MB to achieve 60%, 70%, 80%, and 90% load factors. For labeling, the order of the 1st k-mer in each block was stored in the external labeling space (See the methods section). MPHFs were constructed with sizes ranging from 15 to 22 MB to fit the four datasets. All data structures were queried with 35M existing k-mers and the query times were reported. The MQFs were 10 folds

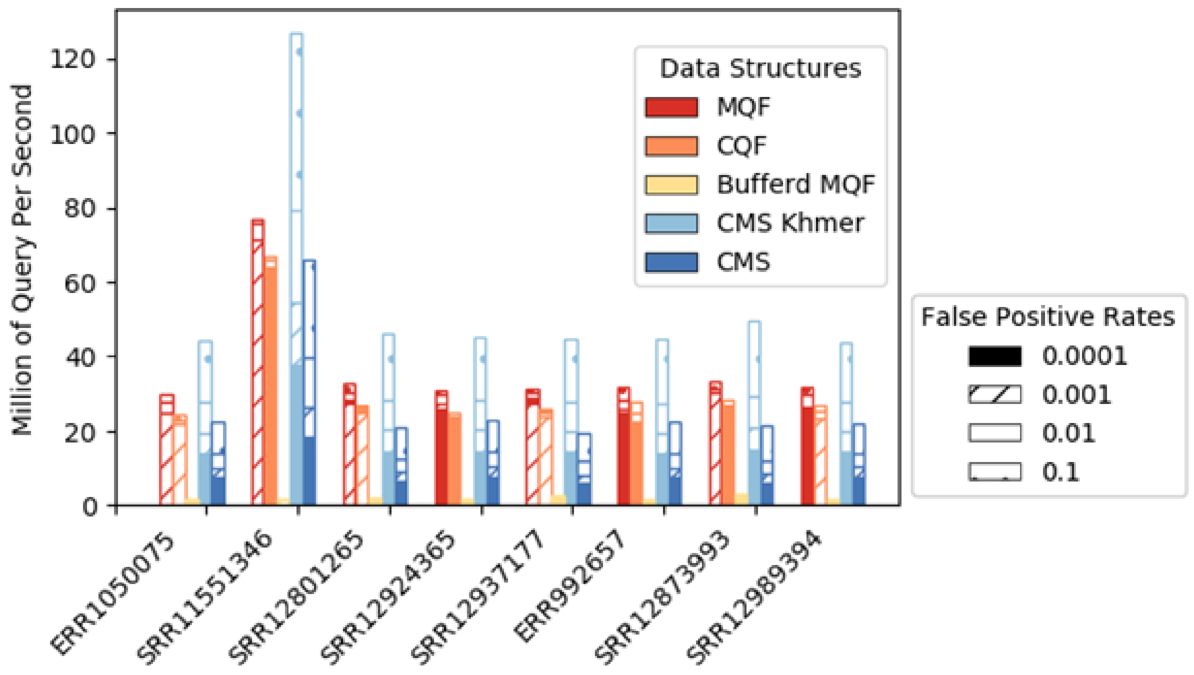

Figure 2.8. **Query rate comparison of four data-structures in real datasets:** MQF, CQF, buffered MQF (using 1/3 the size of other structures), Khmer implementation of CMS, and original implementation of CMS. The performance is plotted as a bar graph where the pattern of each bar shows the cumulative increase in queries for different false positive rates.

faster than the MPHFs. The query time of the MQF was invariable over the different load factors (Figure 2.9).

## 2.3 Discussion

MQF is a new variant of counting quotient filters with novel counting and labeling systems. The new counting system increases memory efficiency as well as the speed of insertions and queries for a wide range of data distributions. The labeling system provides a flexible framework for labeling the member items while maintaining good data locality and a concise memory representation.

MQF is built on the foundation of CQF. MQF has the same ability to behave as an exact or approximate membership query data structure while tracking the count of its members. The insertion/query algorithm developed for CQF enables this family of compact hashtables to perform fast under high load factor (up to 95%)[29]. CQFs are

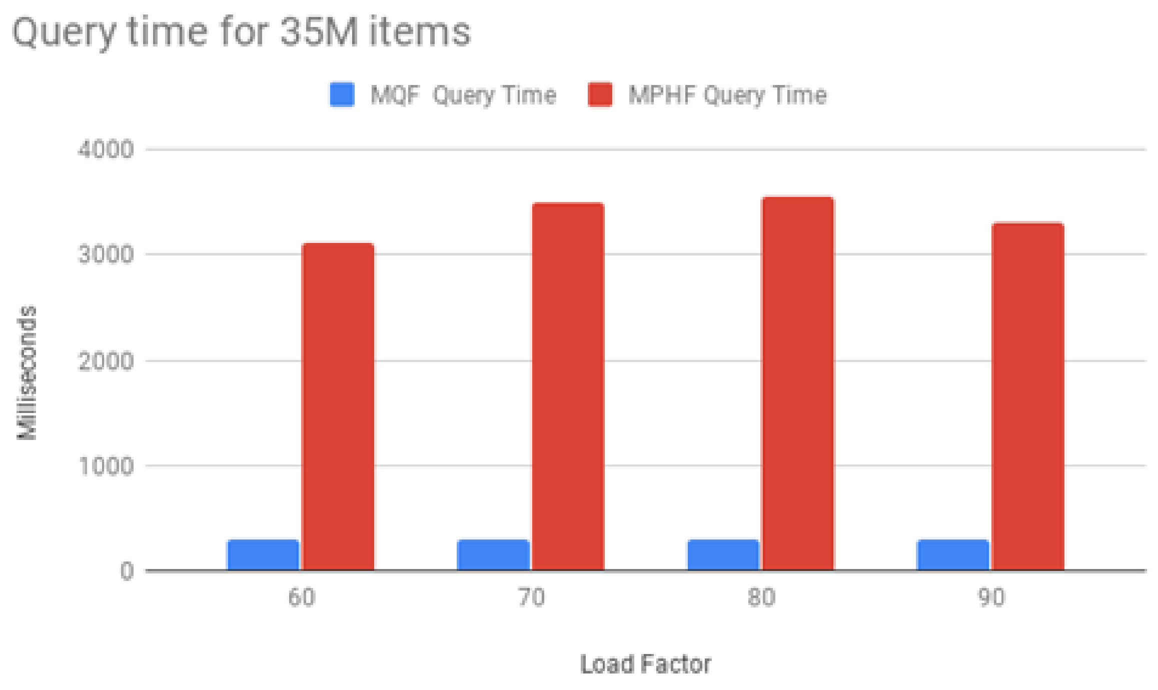

Figure 2.9. **Query performance Comparison of MQF and MPHF.**The query times of 35M existed k-mers were measured for MQF structures with 60%, 70%, 80%, and 90% load factors and MPHF structures storing matching datasets.

designed to work best for data from high Zipfian distributions. However, previous k-mer spectral analysis of RNAseq datasets showed substantial deviations from a Zipfian distribution in thousands of samples[31]. Such variations in distribution are expected given the variety of biosamples, the broad spectrum of sequencing techniques, and different approaches to data preprocessing.

MQF implements a new counting system that allows the data structure to work efficiently with a broader range of data distributions. The counting system adopts a simple encoding scheme that uses a fixed small space alone or with a variable number of the filter's slots to record the count of member items (Figure 2.10). Items with small counts utilize the small fixed-size counters. Therefore, slots, used to be consumed by CQF as counters for these items, are freed to accommodate more items in the filter. The MQF's load factor grows slower than CQF with all distributions except the extreme Zipfian case (Z=5) where the load factor is almost the same (Figure 2.1). This is why the memory requirement for MQFs is usually smaller compared to CQFs under most distributions despite the extra space taken by the fixed counters (Figure 2.2). The size of the fixed-size counter is constant independent from the slot size, therefore the memory requirement for this counter will be trivial with big slots for smaller FPRs and almost negligible in the exact mode. However, this fixed-size counter comes with an additional advantage for MQF. Tuning the size of the fixed-size counter enables the filter to accommodate more items with a slightly larger slot size. This allows the memory requirement for MQF to grow gradually instead of the obligatory size doubling seen in CQF (Figure 2.2-2.5).

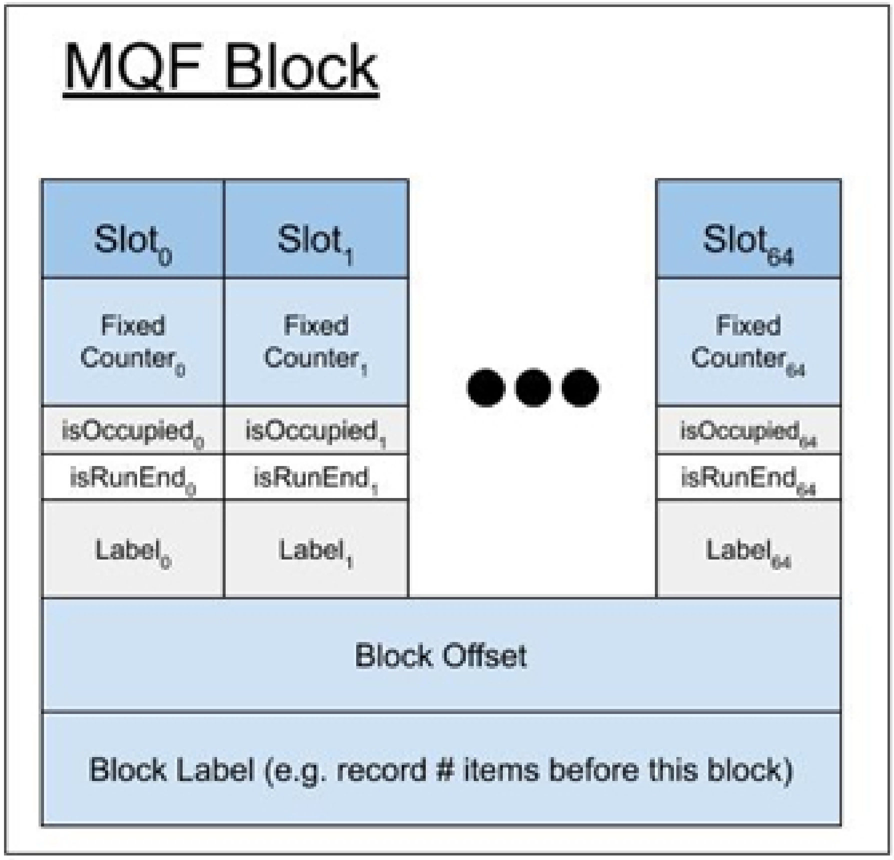

Figure 2.10. **MQF block structure.** Each MQF block contains 64 slots with their metadata, a one-byte block offset, and configurable size space to hold the number of items inserted in the filter before the current block. The metadata of each slot consumes r bits, one bit for each isOccupied and isRunEnd metadata, and configurable f-bits and t-bits for the fixed counter and the slot-specific label respectively.

Moreover, the new counting scheme in MQF is simplified compared to that of the CQF. MQF defines the required memory for any item based solely on its count (Figures

2.11-2.12). Therefore, an accurate estimation of the required memory for any dataset can be done extremely quickly by an approximate estimation of data distribution[43, 44]. This is unlike CQF which needs to add a safety margin to account for the special slots used by the counter encoding technique since it is impossible to estimate the number of these slots.

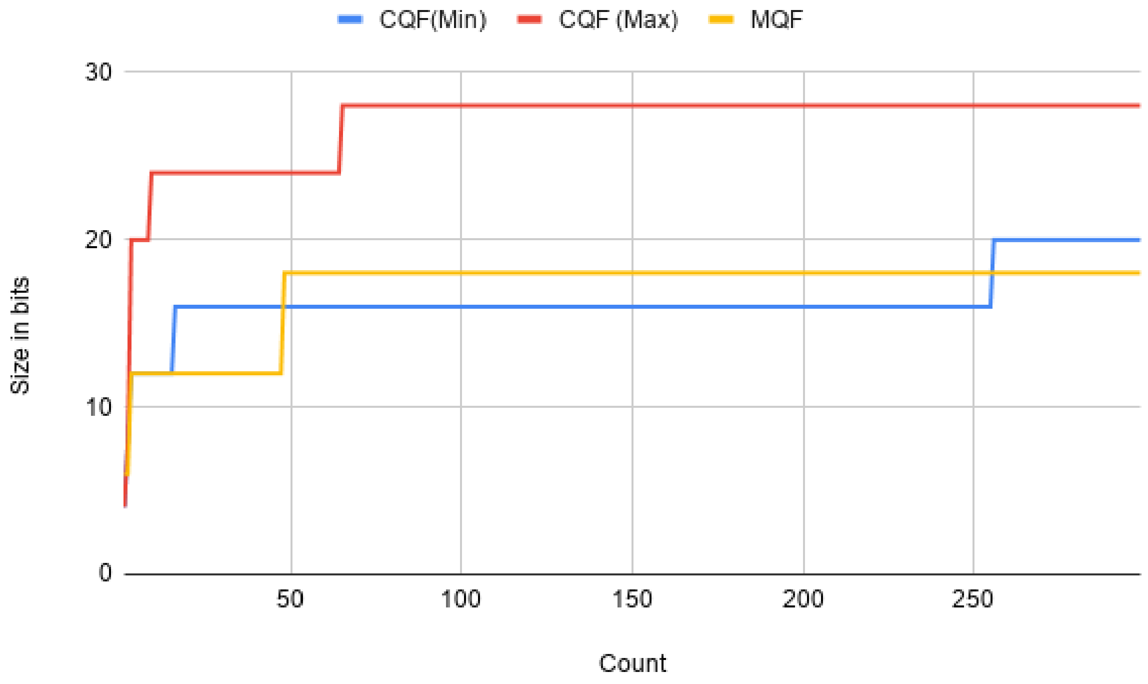

Figure 2.11. **Number of slots required by CQF and MQF** The number of slots needed by each data structure to represent a k-mer and its count. Since CQF needs to add special slots for some items CQF(Min) and CQF(MAX) curves are added to represent the minimum and the maximum number of slots needed by CQF. In this experiment, both data structures are using 4-bit slot size. The MQF slot size includes the fixed-size counter.

Regarding the speed of insertions and queries, MQF is slightly faster than CQF (Figure 2.3). This could be explained partially by the lower load factor of MQF and partially by the simplicity of the coding/decoding scheme of its counting system. Both MQF and CQF are faster than CMS unless the target FPR is really high (e.g. FPR >0.1) (Figure 2.3). CMS controls its FPR by increasing the number of its hash tables requiring more time for insertions and queries to happen. In comparison, quotient filters use always one function but with more hash-bits to control the FPR, with a minimal effect on the

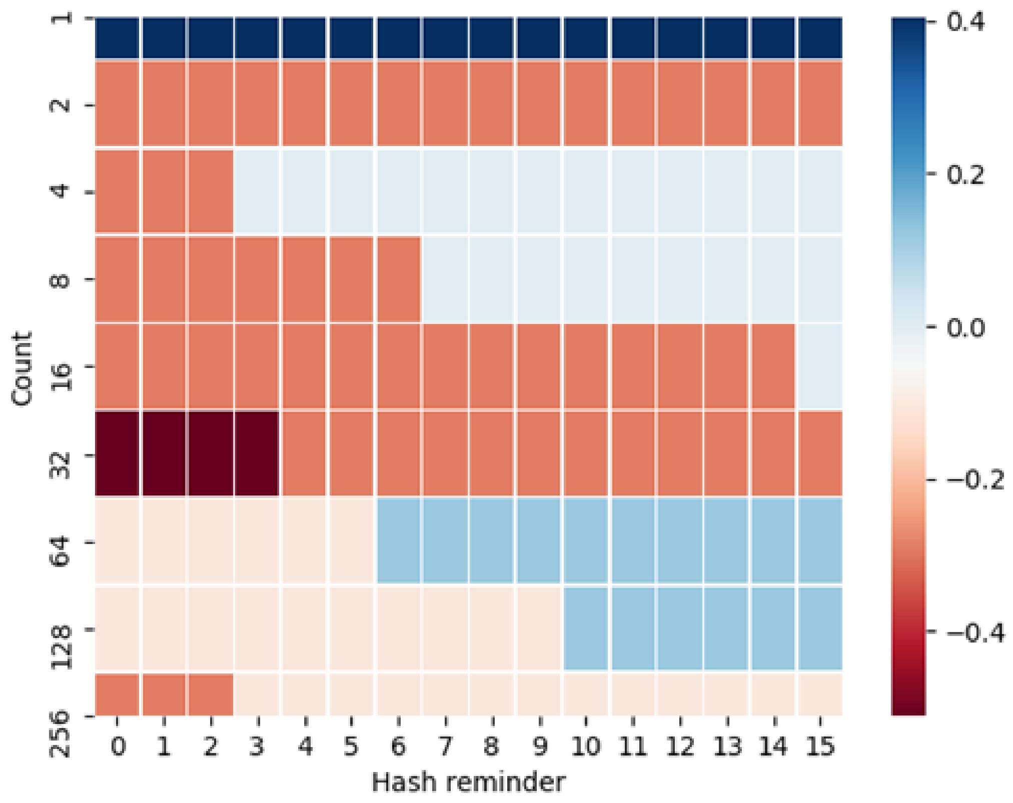

Figure 2.12. **A comparison between the number of slots required by CQF and MQF to represent a k-mer and its count.** A dataset of 144 items was created in the form of k-mer and its count. All possible values of hash reminders of size 4 (16 items) are created with counts in the range starting from 1 to 256. Items were inserted into both data structures and the number of slots needed was recorded. Each cell in the figure displays the ratio of memory requirements on the log scale calculated as Log(((slot size * #slots) in MQF)/((slot size * #slots) in CQF)) represented as a heat map where reddish shades mean that MQF needs fewer slots than CQF.

insertion/query performance (Figure 2.3). With high FPR (e.g. FPR = 0.1), CMS uses fewer hash functions and is better performing than MQF. A quotient filter or CMS with a FPR = δshould have the same probability of item collisions. However, the quotient filter will be more accurate because CMS has another type of error with a probability (1-δ), which incorrectly increases the count of its items. This error is a “bounded error” with a threshold that inversely correlates with the width of the CMS[32]. In another sense, some applications might deploy CMS with a smaller table’s width to be more memory efficient than MQF if the application can tolerate a high bounded error.

Buffered MQF can trade some of the speed of insertions and queries for significant memory reduction by storing data on disk. The buffered structure was developed to make use of the optimized sequential read and write on SSD. The buffered structure processes most of the insertion operations using the bufferMQF that resides on memory, thereby limiting the number of access requests to the MQF stored on the SSD hard drive. Sequential disk access happens when the bufferMQF needs to be merged to the disk. This approach is very efficient for insertions but not for random queries which require more frequent SSD data access. Therefore, bufferMQF is best used for the applications where insertions represent the performance bottleneck e.g. k-mer counting applications5 and sequencing dataset indexing[45]. Moreover, in k-mer analysis of huge raw datasets, buffered MQF can be used initially to filter out the low abundant k-mers (i.e. likely erroneous k-mers), then an in-memory MQF holding the filtered list of k-mers could be used for subsequent application requiring frequent random queries. This allows multistage analyses where a first pass eliminates likely errors to minimize the memory requirements of computationally demanding applications like in the case of widely used graph-based algorithms[46, 47]

CMS is commonly used for online or streaming applications as long as their high error rate can be tolerated[48]. MQF has a better memory footprint in the approximate mode for lower error rates and thus can compute with CMS for online applications. A major advantage of quotient filters compared to CMS is the dynamic resizing ability in response to the growing input dataset[29]. The buffered version of MQF can be very useful when the required memory is still bigger than the available RAM. We should, however, notice that online applications on MQF cannot make use of the memory optimization that could be achieved with an initial estimation of the filter parameters. A new version of the Khmer software that replaces CMS with MQF proves the new data structure more efficient in real-life applications. The MQF version is faster than the one with CMS unless the target FPR is high. Also, MQF is always more accurate than CMS although both structures have the same FPR. This behavior of CMS is due to the high error bound of its counts.

Unlike CQF, MQF is designed to be a more comprehensive associative data structure. MQF comes with a novel labeling system that supports associating each k-mer with multiple values to avoid redundant duplication of k-mers' keys in separate data structures. There are two types of labels: Internal labels adjacent to each item to achieve the best cache locality by storing labels next to the k-mer. However, it has a fixed size and thus practically useful when a small size label is needed. The second labeling system is to label the k-mers with one or more labels stored in external arrays while using the k-mer order in the MQF as an index. External labeling is very memory efficient mimicking the idea of the minimal perfect hash function (MPHF)[39, 40]. MPHF undoubtedly has the least memory requirement of all the associative data structures[40] However, MQF has better performance in both the construction and query phases. For construction, both structures require initial k-mer counting. MQF needs just an extra O(N) operation to update the block labels where N is the number of its unique k-mers. MPHF has to read then rehash the list of unique k-mers possibly more than once which makes it slower than MQF. For query operations, MQF is 10x faster regardless of the load factor of MQF (figure 2.9).

Furthermore, MQF offers more functionality and has fewer limitations than MPHF. MQF is capable of labeling a subset of its items which saves significant space for many applications. For example, k-mer analysis applications may want to only label the frequent k-mers, as an intermediate solution between pruning all the infrequent k-mers and labeling all the k-mers. Moreover, MQF allows online insertions and deletions of items as well as merging of multiple labeled MQFs (See the methods) while MPHF - which doesn't store the items - needs to be rebuilt over the whole dataset, which requires reading and rehashing the datasets. Furthermore, MQF can be exact, while MPHF has false positives when queried with novel items that don't belong to the indexed dataset.

## 2.4 Conclusion

MQF is a new counting quotient filter with a simplified encoding scheme and an efficient labeling system. MQF adapts well to a wide range of k-mer datasets to be more memory and time-efficient than its predecessor in many situations. A buffered version of MQF has a fast insertion algorithm while storing most of the structure on external memory. MQF combines a fast access labeling system with MPHF-like associative functionality. MQF performance, features, and extensibility make it a good fit for many online algorithms of sequence analysis.

## 2.5 Methods

**MQF Data structure:** MQF has a similar structure to CQF with a different scheme of metadata that enables different counting and labeling systems (Figure 2.10). Like the CQF, the MQF requires 2 parameters, r and q, and creates an array of $2^{qslots}$; each slot has r-bits. In MQF, $Q_i$ is the slot at position i where i = 1 … $2_q$. The MQF maintains the block design of CQF where each block has 64 slots with their metadata and one extra byte of metadata called Offset to enhance the query of items[29]. Both MQF and CQF have two metadata bits to accompany each slot: $isRunEnd_i$ and $isOccupied_i$. In the MQF, each slot i has extra metadata, a fixed-size counter with a value ($F_i$) and a configurable size ($F_{size}$). There are also two optional fixed-size parts of metadata allocated to allow different styles of labeling. Every slot has specific labeling ($ST_i$) with a configurable size ($ST_{size}>= 0$), and every block ( j ) has an optional space of a configurable size designed to store the number of items in the previous blocks.

The MQF uses the same insertion/query algorithm of CQF[29]. In brief, suppose item I, repeated c times, is to be inserted into Q. A hash function H is applied to I to generate a p-bit fingerprint (H(I)). H(I) value is split into two parts, a quotient and remainder. The quotient ($q_i$) is the most significant q bits while the remainder ($r_i$) is the remaining least significant r bits. The filters store $r_i$ in a slot $Q_j$ where where j is determined by linear probing stating from $q_i$. One or more slots can be used to store the count of the same item. If the required slots for the item or its count are not free, all the consecutive occupied slots starting from this position will be shifted to free the required space. All items having the same q are stored into consecutive slots and are called a run. Items in the run are sorted by $r_i$, and isRunEnd of the last slot in the run is set to one. isOccupied ($q_i$) is set to one if and only if there is a run for $q_i$. Therefore, there is one bit set to one in each isOccupied and isRunEnd for each run. To query item I, a Rank and Select method is applied on the metadata arrays to get the run start and end for $\mathrm{q_i}$. Then all the items in the run are searched linearly for the slot containing $\mathrm{r_i}$. The subsequent one or more slots can be decoded to get the count of item I. CQF uses a special encoding scheme to recognize these counting slots but MQF utilizes the fixed-counter metadata element (see below).

### 2.5.1 Counting Scheme

MQF uses variable-length integers to encode the k-mers count as shown in figure 2.13. It uses as many slots as required to encode the count while using the fixed-size space of the slots to mark the last slot. To do so, each fixed-size space in all slots before last is assigned its max value ($\mathrm{F_{max}}$), while the fixed-size space of the last slot stores the most significant bits of the count ($\mathrm{F_i}$) where ($\mathrm{F_i}$¡$\mathrm{F_{max}}$). If the value of the most significant bits is equal to $\mathrm{F_{max}}$ , an extra slot with zero bits will be added (see example 3 in Figure 2.13). For an item with count c, the fixed-size space of the item's slot is enough for counting until C /textgreater= $\mathrm{F_{max}}$ where the number of required slots for counting can be calculated as $Ceiling(\frac{abs(\log_2(c-f_{max})-f_{size})}{r}) + Floor(\frac{c_s}{f_{max}})$ where $\mathrm{c_s}$ represents the decimal value of the most significant bits.

In comparison to the CQF, the MQF does not use special slots to resolve ambiguities, which is more memory efficient. The counter encoding algorithm is described in Figure 2.14.

### 2.5.2 Parameter Estimation

For offline counting applications, the MQF parameters (q, r, $\mathrm{F_{size}}$) can be even more optimized for each dataset to create the most memory-efficient filter that has enough slots to fit all unique items and their counts. The q parameter defines the number of slots (N) in MQF where $q = \log_2(N)$. The required numbers of slots for items and their count can be estimated from the cardinality of the target dataset, as with CQF. The r parameter is calculated from the equation r = p-q where p is the total number of hash-bits used to represent each item. In the exact mode, p equals the exact output of a reversible

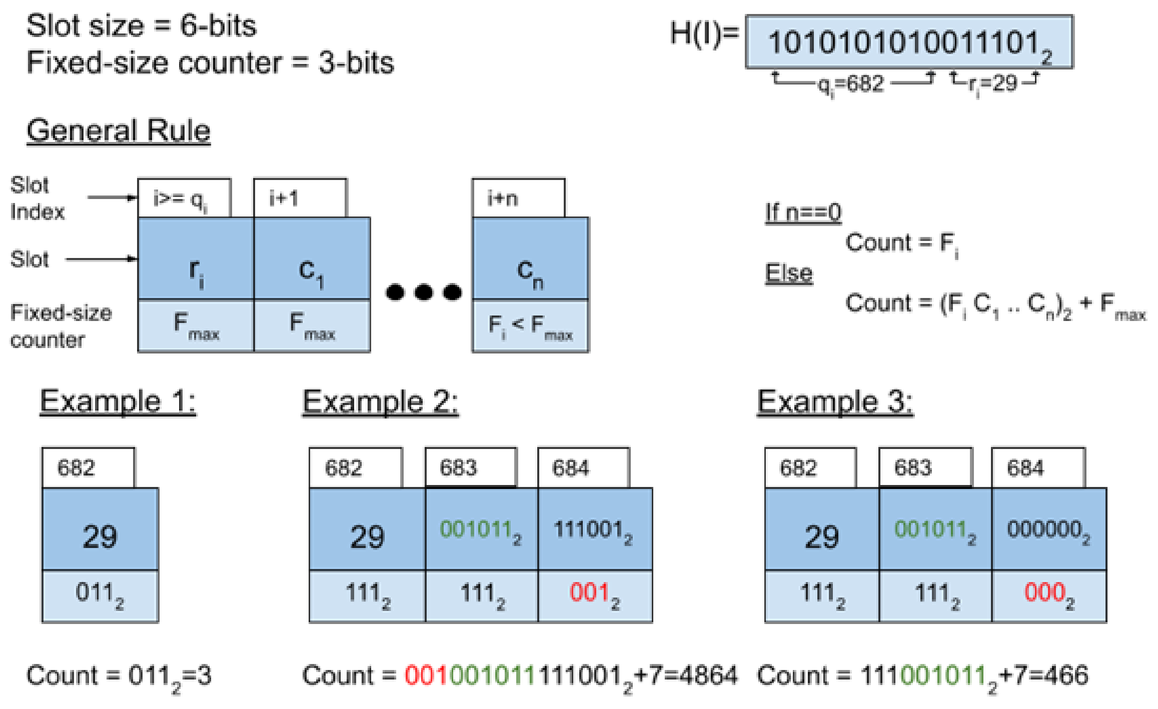

Figure 2.13. **MQF counters encoding scheme.**Items and their counts are stored in n slots and n fixed counter as shown in the general rule. Each example stores the same item but different count (count = 3, 4864, or 466).

hash function. In the inexact mode, p is controlled by the target FPR ($\delta$) according to the equation $q = \log_2(\frac{N}{\delta})$ described before[29]. The F$_{\text{size}}$ parameter defines the size of the fixed-size counter. This is critical because if a given MQF has too few slots for items in a dataset, the bigger MQF would have to double the number of slots causing a big jump in the memory requirement. To avoid that jump, MQF can use larger fixed size counters to decrease the number of slots required in counting on the expense of a slight increase in the slot size. MQF comes with a utility script (https://github.com/dib-lab/MQF/blob/master/include/utils.h) to enable the calculation of the optimum parameters.

## 2.5.3 Labeling System

MQF can map each item to its count as well as other values, which we call "labels". Labels in MQF have two different systems. An internal labeling system stores the associated value for every key in the data structure, like a hash table. This label has a fixed size defined

**Algorithm 1** Counters Encoder

```
procedure ENCODE($Q$, $r_i$, $count$ ,$start$)
    $base \leftarrow 2^{Q.r}$                                ▷ Q.r is #bits in the slot
    $fcountMax \leftarrow 2^{Q.f} - 1$                       ▷ Q.f is #bits in the fixed-size counter
    $stack \leftarrow \phi$
    while $count > fcountMax - 1$ do
        $stack.push(count\%base)$
        $count = count >> Q.r$                               ▷ bit shift operation
    end while
    $stack.push(r_i)$
    $i \leftarrow start$
    while $stack \neq \phi$ do
        $Q_i.r \leftarrow stack.pop()$                       ▷ Slot of index i
        $Q_i.f \leftarrow fcountMax$                         ▷ fixed-size counter of index i
        $i \leftarrow i + 1$
    end while
    $Q_{i-1}.f \leftarrow count$
end procedure
```

Figure 2.14. **Counters encoder algorithm.**The Algorithm encodes the item and its count into one or more slots. The first slot is reserved for the item's remaining, and the variable number of slots follows to encode the item's count.

at the initialization of the MQF and is practically useful when a small size label is needed (e.g. one or two bits). The second labeling system labels the block. We use this label to store the number of items inserted in the MQF before each block. This enables labeling the items of the filter by separate arrays matching the order of the items in the filter, a behavior that can act as a minimal perfect hash function[40]. The naive way to compute the items' order is to find the item in the MQF and iterate backward until the beginning of the filter to count the number of the preceding items, which is an O(N) operation. The MQF stores the number of items that exist before each block; therefore, the MQF iterates only to the beginning of each block, which is an O(1) operation. The number of previous items for each block is computed after the MQF is constructed. Any additional insertions or deletions of items would only require re-calculation of the block label values with no need to re-analyze the original data. Moreover, labeled MQFs can be updated by merging multiple labeled MQFs and their external labeling arrays. External label arrays need to be merged after merging the labeled MQFs. To do so, the new items' order is recomputed in the final MQF. Then, labels in the input external arrays can be copied into a new external array according to the new item order. Such a function has to consider resolving the conflicts of items happening in multiple-input MQF and labeled by different external labels (Figure 2.15).

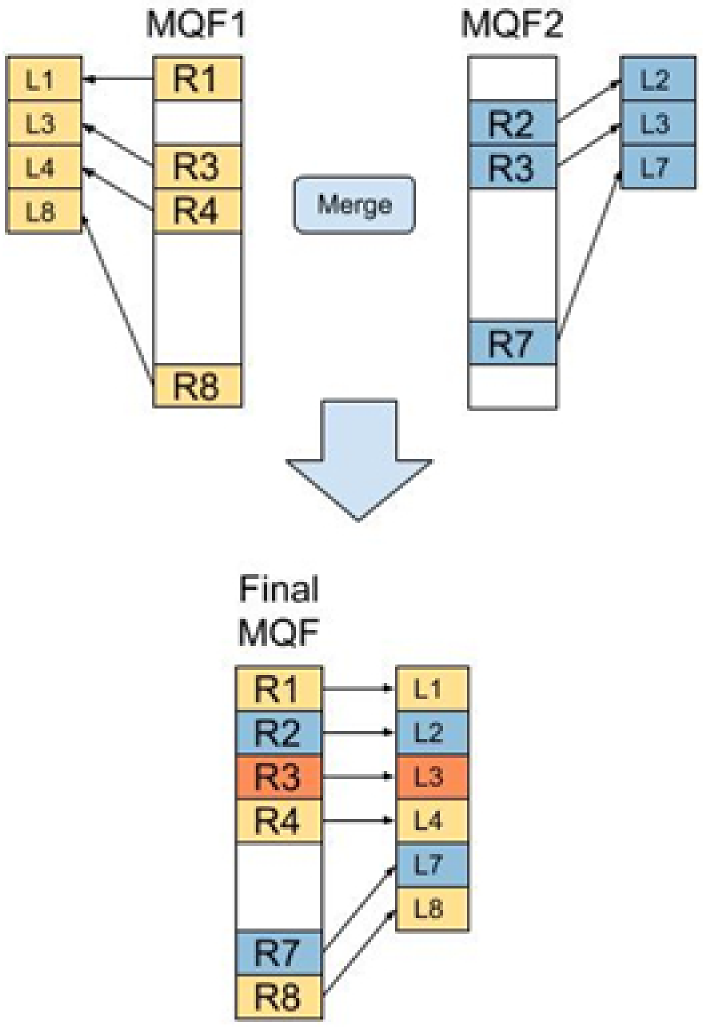

Figure 2.15. **Merging MQFs with external labels.**$R_i$ is the remaining part of item i, and $T_i$ is the external label of the item. Merging the input MQF produces a final MQF with a new order of its member items. All labels in the input external arrays are copied into a new external array according to this new order of the items. However, the implementation of the merge function has to resolve the conflict of R3 labels which exist in both input structures with two labels.

### 2.5.4 Buffered MQF

The Buffered MQF is composed of two MQF structures: a big structure stored on SSD called onDiskMQF, and an insertion buffer stored in the main memory called bufferMQF. OnDiskMQF uses stxxl vectors[49] because of the performance of their asynchronous IO. The bufferMQF is used to limit the number of accesses on the OnDiskMQF and change the access pattern to the on-disk structure from random to sequential. As shown in the insertion algorithm in Figure 2.16, all the insertions are done first on bufferMQF; when it is full, the items are copied from bufferMQF to OnDiskMQF, and bufferMQF is cleared. The copy operation edits the onDiskMQF in a serial pattern which is preferred while working on SSD because many edits will be grouped together in one read/write operation. Figure 2.17 shows the query algorithm. The queried items are inserted first to temporary MQF and sequential access is done to query the items from the OnDiskMQF. The final count is the sum of the bufferMQF and the ondiskMQF.

**Algorithm 2** Buffered MQF Insertion

```
1: procedure INSERT($onDiskMQF$, $bufferMQF$, $item$)
2:     $mqf\_insert(bufferMQF, item)$
3:     if $mqf\_space(bufferMQF) > 90$ then
4:         for all $i \in bufferMQF$ do
5:             $mqf\_insert(onDiskMQF, i)$
6:         end for
7:         $mqf\_clear(bufferMQF)$
8:     end if
9: end procedure
```

Figure 2.16. **Buffered MQF insertion algorithm.** Insertion Algorithm for inserting items in the Buffered MQF. It inserts the item in the in-memory data structure. The on-memory structure is merged into the on-disk structure when it is filled.

### 2.5.5 Experimental Setup of Benchmarking

A total of 13 datasets (5 simulated and 8 real sequence data) were used in the experiments to cover a broad spectrum of fragment selection approaches and sequencing platforms. Three datasets called z2, z3, and z5 were simulated to follow Zipfian distribution using three different coefficients: 2, 3, and 5 respectively. The bigger the coefficient the more

**Algorithm 3** Buffered MQF Query

```
1: procedure QUERY($onDiskMQF$, $bufferMQF$, $list\_item$)
2:     for all $i \in listItems$ do
3:         $mqf\_insert(tmpMQF, i)$
4:     end for
5:     for all $i \in tmpMQF$ do
6:         $counts[i] \leftarrow mqf\_query(onDiskMQF, i)$
7:         $counts[i] \leftarrow counts[i] + mqf\_query(bufferMQF, i)$
8:     end for
9:     return $counts$
10: end procedure
```

Figure 2.17. **Buffered MQF query algorithm.** Query algorithm for retrieving counts for a list of items in the Buffered MQF. First, insert all the items in the list into a temporary MQF. Second, iterate over the list of items in the temporary MQF and query both the in-memory and on-disk structures.

singletons in the dataset[50]. A fourth dataset was simulated from a uniform distribution with a frequency equal to 10. Another dataset was simulated to mimic single-end 100 bp DNA Illumina sequencing from human chromosome 20 using ART simulator[51]. The remaining 8 datasets represent k-mers sampled from real sequencing datasets (ERR1050075, SRR11551346, SRR12801265, SRR12924365, SRR12937177, ERR992657, SRR12873993, SRR12989394). Throughout the manuscript, each dataset is referred to by its accession number. Table2.4 enlists all the datasets with their description. A k-mer size of 25 nucleotides was used in all experiments.

| Sample Accession | Description |
|---|---|
| ERR1050075 | PolyA selected mRNA sequencing of Homo sapiens |
| SRR11551346 | Whole Exome sequencing of Homo sapiens |
| SRR12801265 | 3' mRNA-Seq - Mouse B cell lymphoma |
| SRR12924365 | Single-cell sequencing RNA of Mice |
| SRR12937177 | RAD-Seq with Ion Torrent for Cucurbita argyrosperma subsp |
| ERR992657 | DNA Whole-genome sequencing |
| SRR12873993 | RNA immunoprecipitation sequencing (RIP-Seq) - human cells infected with Ebola virus |
| SRR12989394 | DNase-seq of mitochondrial genome of Mukaria Splendida |

Table 2.4. **Description of all the samples used in MQF chapter.**

Experiments were conducted to compare the performance, memory, and accuracy of MQF with the state-of-the-art counting structures CQF, CMS, and MPHF. Unless stated otherwise, CQF and MQF used the same number of slots, and the same slot size while the fixed counter of MQF was set to two. The slot size was calculated to achieve the target FPR as described in the parameter estimation section (see Methods). To create comparable CMS, the number of the tables in the sketches was calculated using $ln(\frac{1}{\delta})$ as described before[32]. The table width was calculated by dividing the MQF size by the number of tables. The MPHF was created using the default options in the BBhash repo (https://github.com/rizkg/BBHash). An Amazon AWS t3.large machine with Ubuntu Server 18.04 was used to run all the experiments. The instance had 2 VCPUS and 8GB RAM with a 100GB provisioned IOPS SSD attached for storage. All codes used in the experiments can be accessed through the MQF GitHub repository (https://github.com/dib-lab/2020-paper-mqf-benchmarks).

## 2.6 List of abbreviations

- MQF: mixed-counters quotient filter.
- CQF: counting quotient filter.
- FPR: false positive rate.
- CMS: count-min sketch.
- MPHF: Minimal Perfect Hash Functions

# Chapter 3

# The Great Genotyper

## 3.1 Introduction

Maya Angelou once beautifully articulated, 'In diversity, there is beauty and there is strength.' This concept is strikingly relevant in genomic studies, highlighting the necessity of investigating genetic diversity across large cohorts and populations to enhance our understanding of evolution [52, 53], genetic adaptations [54], and gene-disease associations [55, 56]. Genome diversity arises from various types of variations: single nucleotide variants (SNV), small insertions and deletions (less than 50bp), and structural variants (larger than 50bp). Structural variants (SVs) contribute to genomic diversity at a rate 15 times greater than SNVs [57] and have a more pronounced impact on gene function[33]. Yet, SVs are less studied than smaller variants because the abundant short-read sequencing (SRS) methods have a high false positive rate. Conversely, Long-read technologies are more accurate but expensive for the population scale. Investigating SVs in populations can deepen our understanding of genomics and offer insights into ourselves and other organisms.

Variant calling from long read sequences (LRS) offers greater reliability in terms of precision and recall [6], using mapping [13, 58, 59] or assembly-based approaches [60]. Datasets of LRS are on the rise as the technology becomes more affordable. For example, the Human Pangenome Reference Consortium (HPRC) is increasing the number of samples from 47 to 350 [61], and Chinese researchers [62] developed a pangenome from

58 samples of 36 distinct Chinese ethnic minority groups. The number of pangenomes for other organisms is also growing substantially [20, 63, 64, 65]. However, the amount of LRS data available still pales in comparison to SRS. Consequently, there is an urgent need to develop computational techniques that leverage the high-quality variant discovery of LRS while capitalizing on the abundance of SRS.

Short-read alignment approach for detection of SVs yields a low and unreliable recall rate, ranging from 10% to 70% [6]. Moreover, mapping reads to a linear genome reference introduces bias towards the subpopulation predominantly represented in the reference genome assembly, such as the European subpopulation in GRCh38 [17, 18, 19]. Additionally, merging SV callsets from different individuals presents its own set of challenges. The same SV can display shifted coordinates across different samples [12]. In essence, the inherent limitations of calling SV from SRS complicate studying SV on a population scale.

To address the shortcomings of SV callers, dedicated genotypers examine the presence and genotype of SVs -whether called from SRS or LRS- in SRS samples [24, 66, 67, 68, 23]. For instance, Paragraph [66] and Graphtyper2 [68] realign the reads to a variation-aware graph and determine the genotypes from the realignment, thus mitigating the mapping bias. Pangenie [23] was developed to genotype phased variants from pangenomes using kmers specific to all possible alleles to overcome the mapping bias entirely. Uniquely, Pangenie integrates both genotyping and imputation simultaneously, leveraging the phasing information of the variants in the pangenome to infer genotypes in regions with absent coverage. As a result, Pangenie achieves superior outcomes compared to other SV genotypers.

Genotyping SVs yields higher recall and precision than detecting those variants by directly calling them in the same SRS samples. For instance, Huddleston et al. [69] employed LRS to analyze SVs in two human genomes. They found that 90% of these SVs were missing in the 1000 Genomes callset. Yet, 61% of these SVs could be genotyped using short-read sequencing. Therefore, many recent population-scale studies adopted a joint approach of variant calling and genotyping. Initially, variants are called from a small number of LRS samples or a larger number of SRS samples. Subsequently, the discovered SVs are merged and genotyped in a larger cohort of SRS [16]. For instance, Kirsche et al. [70] employed Paragraph [66] to genotype variants detected from 31 LRS samples in a population of 1.3k SRS samples from the the 1000 Genome Project (1kGP) [71]. Similarly, Graphtyper2 [68] was used to build a graph using SVs detected in SRS of 50k Icelandic samples [68] or 2k dogs [15]. The graphs were then re-genotyped with the same SRS samples to enhance recall. Conversely, muCNV leverages population information to refine genotyping, by modeling the read mapping statistics across a population of samples to enhance the genotyping accuracy. Goo Jun et al. [14] employed MuCNV to jointly genotype TopMed SV structural variants across 139k SRS samples. Lastly, The HPRC [61] utilized Pangenie to genotype pangenome variants in 3,202 samples from 1kGP [71].

Population genotyping methods, although swift and scalable, do not address the N+1 problem. Introducing a single new variant necessitates downloading and reprocessing all the raw data, which is unfeasible for most labs. Current approaches are not efficient enough to keep up with the rapid generation of pangenomes and population-specific variant catalogs. Genotyping these variants in the available SRS can elevate population genetics to new heights. The SV N+1 challenge also impacts disease gene discovery studies in probands [2]. LRS can produce phased, high-quality SVs, and finding pathogenic variants requires filtering out common variants and limiting the search scope to rare variants. However, matching these variants to public databases is challenging, and the reliability of allele frequencies in SV catalogs like genomeAD [72] is questionable if they were calculated in small cohorts, in different subpopulations, and/or using low recall techniques. Genotyping new variants in a large population, covering all subpopulations if possible, provides a more accurate and informative AF calculation.

Shifting gears in population genomics, we present "The Great Genotyper", an alignment-free genotyping pipeline for both SV and small variants. We show how it can genotype four thousand human samples from different subpopulations in mere hours, without the need for 95TB of raw sequences . Instead, it uses a Counting Colored Debruijn Graph (CCDG) of size 1TB. The CCDG needs to be created once from the raw sequences of a given population by retaining the kmers and their counts in a compressed form. The CCDG offers a realistic solution for the N+1 problem because it can be re-used for genotyping any new list of variants in this population . In addition to unprecedented performance, The Great Genotyper's genotyping accuracy is similar to the state of the art since it uses the Pangenie genotyping model for phased variants and improves the genotyping quality with population derived information. Moreover, The Great Genotyper applies the Pangenie model on unphased variants by initially phasing them using the CCDG. Therefore, we were able to demonstrate how to merge different gene catalogs with pangenome-derived variants in one phased panel of 4k human samples. The panel was utilized for fine mapping of GWAS peaks and imputation of common SVs.

## 3.2 Background

The Great Genotyper stands on the shoulders of state-of-the-art methods to deliver both scalable and accurate population genotyping. It harnesses the power of Metagraph for the creation and querying of Counting Colored de Bruijn graphs. Also, it employs Pangenie's genotyping model for single sample genotyping, and leverages Beagle for imputation and phasing. In this section, we delve into the key tools that have been instrumental in propelling The Great Genotyper to its unparalleled levels of performance and precision.

### 3.2.1 Metagraph

Genomic sequences can be represented using De Bruijn graphs, where each k-mer is a node and an edge is formed if the k-mers representing the two nodes share k-1 bases. An evolution of the standard De Bruijn graph is the Colored De Bruijn Graph, which can represent multiple samples by conceptually associating each sample with distinct colors [73]. The process of creating a colored De Bruijn graph is often referred to as k-mer indexing [74, 45]. While Colored De Bruijn Graphs (or k-mer indexes) merely track which samples contain each k-mer, Counting Colored De Bruijn Graphs (CCDG) also record the k-mer abundance for each sample, paving the way for a wider range of applications. Nonetheless, constructing a CCDG for thousands of samples is a formidable challenge, demanding efficient algorithms and data structures.

Metagraph [75] and REINDEER [76] provide succinct data structures to house the CCDG. Both tools harness the principle that related k-mers often share similar metadata, thereby building the index around the dataset's unitigs. For our purposes, we opted for Metagraph to generate a CCDG for 4,271 samples that represent the global populations. This CCDG encapsulates k-mers from all the samples along with their respective counts. These k-mers and their counts are subsequently leveraged by The Great Genotyper during the population genotyping process.

The indexing algorithm of Metagraph begins with k-mer counting using KMC [77] for each sample. Following this, a graph is constructed based on the KMC counts. This allows Metagraph to apply optional error-cleaning techniques inspired by mccortex [78]. These cleaning algorithms tackle three primary types of errors:

1. They eliminate tips in the graph smaller than twice the k-mer size. Here, "tip" refer to unitigs connected from only one end in the graph.

2. They discard unitigs with k-mer abundance falling below a certain threshold. This threshold can be automatically determined based on the data.

3. K-mer counts are smoothed by retaining the average k-mer count for the unitig as opposed to individual k-mer counts.

These strategies significantly reduce the memory footprint of the final CCDG while sacrificing minimal information. After error cleaning of individual graphs, Metagraph constructs a combined graph by merging all individual graphs. Subsequently, it appends each sample's k-mer counts to this composite graph. The k-mer counts undergo further optimization by smoothing to minimize their size in the data structure. Ultimately, the resulting graph can be several hundred times smaller than the initial raw data, preserving most of the genuine k-mers and their respective counts.

### 3.2.2 Pangenie

Pangenie genotype phased small and structural variants in SRS samples using a k-mer-based method. This involves extracting a set of k-mers unique to the variants targeted for genotyping. The genotyping and imputation processes leverage a Hidden Markov Model, inspired by the Li-Stephen imputation model [79]. This model utilizes the k-mer counts from the extracted unique k-mers, and it imputes genotypes for variants lacking k-mer support if k-mer support is found for a proximate variant on the same haplotype. I will elucidate the workings of the Pangenie software, beginning with its method for unique k-mer extraction and the criteria behind this. Subsequently, I will delve into the Hidden Markov Model employed for genotyping and imputation.

While it may initially seem counter-intuitive to use kmers in genotyping due to the loss of information about the context of kmers in the genome, tools like Nebula [24] and Pangenie [23] have leveraged the fact that variations create new unique kmers that are not found elsewhere in the genome. For instance, a Single Nucleotide Polymorphism (SNP) can give rise to 31 new 31-mers, some of which are exclusive to specific regions in the genome. Similarly, the breakpoints of deletions and insertions generate additional unique kmers, with inserted sequences contributing many specific kmers that can tag the insertion.

Nebula and Pangenie employ distinctive methodologies to identify these variant-specific kmers, omitting kmers present in other regions, and these are referred to as unique kmers.

Extracting unique kmers begins by generating a kmer list from the reference genome in the variation region to represent the wild type. Subsequently, the reference sequence is altered to recruit the variation, and kmers are extracted from these mutated sequences. Reference and mutated kmers are then filtered to remove those present in other genomic regions to mitigate inaccurate signals.

Following this, we count the kmers in the refined list of kmers to obtain the kmer counts from SRS. This kmer list is further refined by discarding kmers with counts exceeding double the sample average. These kmers are unreliable as they may originate from regions duplicated in the sample genome or from regions that are collapsed in the reference genome. Duplicated regions in the sample genome can lead to inflated kmer counts, making it harder to accurately determine the correct genotype from the kmer counts. On the other hand, collapsed regions in the reference genome can falsely suggest a kmer is unique, causing it to be retained when it actually exists in other parts of the genome and should have been filtered out in the previous filtration step.

The Hidden Markov Model genotypes and impute a big chunk of variants at once. To define the Model, assume $H$ is the number of haplotypes in the reference panel. Given a list of variants $V$ of size $v_{size}$, with $V_i$ as the $i^{th}$ variant, we create a set of nodes $N_{v,l,m}$ for each $V_v$, where $l, m \in [0, H]$, representing all possible phased genotypes. Each node $N_{v,l,m}$ connects to all nodes of $N_{v-1,l,m}$ and $N_{v+1,l,m}$, barring the first and last nodes which have connections on one side only.

**Emission Probability**

The Emission probabilities for each node are determined by calculating the probability of the genotype representing the node given the counts of the unique kmer found in the sample.

Let us assume $H_{v,l}$ represents the sequence of Haplotype $l$ at the $v$th Variant. Additionally, let's define $K_v$ as the set of unique k-mers of size K extracted from all the Haplotype sequences $H_{v,l}$, where $v$ ranges from 0 to $v_{size}$ and $l$ ranges from 0 to $H$. The expected copy number of each kmer k, $CN_{v,l,m,k}$, with respect to Haplotypes $l$ and $m$, is calculated using Equation (1).

$$CN_{v,l,m,k} = \left\{ \begin{array}{ll} 0, & K_v[k] \notin H_{v,l} \cup H_{v,m} \\ 1, & K_v[k] \in H_{v,l} \setminus H_{v,m} \\ 1, & K_v[k] \in H_{v,m} \setminus H_{v,l} \\ 2, & K_v[k] \in H_{v,l} \cap H_{v,m} \end{array} \right\} \tag{1}$$

Let's assume the observed count of the $k$th unique kmer, $K_v[k]$, is denoted as $OK_v[k]$. We calculate the expected probability of the observed count given the assumed copy number, $P(OK_v[k] \mid CN_{v,l,m,k})$, as described in Equation (2). Here, $F$ represents a Poisson distribution, where the mean of the distribution is $\frac{\lambda}{2}$ when $CN_{v,l,m,k}$ equals to 1 and 2, respectively. When considering $CN_{v,l,m,k} = 0$, we utilize a geometric distribution with mean $\lambda$. This approach allows for the detection of some k-mers that may occur due to sequencing errors, rather than being solely attributable to the presence of the variant.

$$P(OK_v[k]|CN_{v,l,m,k}) = \left\{ \begin{array}{ll} G(\lambda), & CN_{v,l,m,k} = 0 \\ F(\frac{\lambda}{2}), & CN_{v,l,m,k} = 1 \\ F(\lambda), & CN_{v,l,m,k} = 2 \end{array} \right\} \tag{2}$$

Finally, we compute the emission probability for node $N_{v,l,m}$ using Equation (3). This equation assumes that the kmer counts are independent of each other, and calculates the final probability as the product of all individual probabilities. However, it is important to note that the counts of overlapping kmers are not independent. Integrating the overlap information would overly complicate the algorithm. Therefore, Equation (3) provides an approximation of the true emission probability.

$$Emission(N_{v,l,m}) = \prod_{k \in K_v} P(OK_v[k]|CN_{v,l,m,k}) \tag{3}$$

#### 3.2.2.1 The Transition Probability

The transition probabilities between nodes are computed using the linkage disequilibrium equation from the Li-Stephen model, which increases when $j$ and $k$ are the same for both connected nodes and are closely located in the genome.

#### 3.2.2.2 Genotyping Algorithm

The forward-backward algorithm is then employed to calculate the probability of each node based on the given emission and transition probabilities, and the genotype is subsequently inferred from the node with the highest probability. This refined methodology ensures precise identification and imputation of variants, thereby affirming the reliability and precision of genotyping in phased variants derived from pangenomes.

### 3.2.3 Beagle

Beagle is software designed to phase and genotype variants using a reference panel. It requires a multi-sample VCF file as input, which contains population genotypes for a set of variants. Notably, these genotypes don't have to be comprehensive; Beagle can impute missing haplotypes statistically by modeling the linkage disequilibrium via the Li-Stephens HMM. Moreover, Beagle can phase the genotypes, thereby producing a reference panel suitable for imputing and phasing other samples. In terms of processing time and accuracy, Beagle competes favorably with state-of-the-art solutions and can manage hundreds of thousands of samples efficiently. Owing to its capabilities, Beagle has been utilized to build reference panels using data from projects like UK Biobank and TOPMed [80].

## 3.3 Methods

### 3.3.1 Developing a Counting Colored De Bruijn Graph for a Cohort of Short-Read Sequencing Samples.

Creating a CCDG(also called indexing) for thousands of SRS is challenging, as the size of the raw data surpasses our available resources. Both number of samples and number of unique kmers affect the performance of indexing. Without rigorous kmer removal based on their counts, the number of unique kmers is predominantly driven by errors. However, aggressive kmer trimming may impede genotyping accuracy by incorrectly eliminating low coverage regions, creating a trade-off between performance and final index size and genotyping accuracy. Additionally, metadata are not always reliable due to potential human annotation errors.

To address these challenges, we adopted a strategy of minimal kmer trimming, eliminating kmers with counts fewer than three to preserve genotyping accuracy, and distributing closely related samples across multiple CCDGs. Given the potential annotation errors and heterogeneous sources, metadata is not the optimal solution for identifying related samples.

Figure 3.1 delineates our workflow to balance between genotyping accuracy and managing the constraints of dataset size and resource availability. Also it mitigates the impact of metadata inaccuracies.

Sequencing Data are stored on cloud repositories like SRA in CRAM, FASTQ, or SRA formats. A single 30X human WGS sample, in CRAM format, is approximately 15GB, which can be reduced to around 1.2 GB when stored as unitigs in gzipped fasta format and kmer counts in gzipped format. Therefore, to conserve disk space, data is downloaded and processed in chunks, with the deletion of raw and intermediate files post-processing.

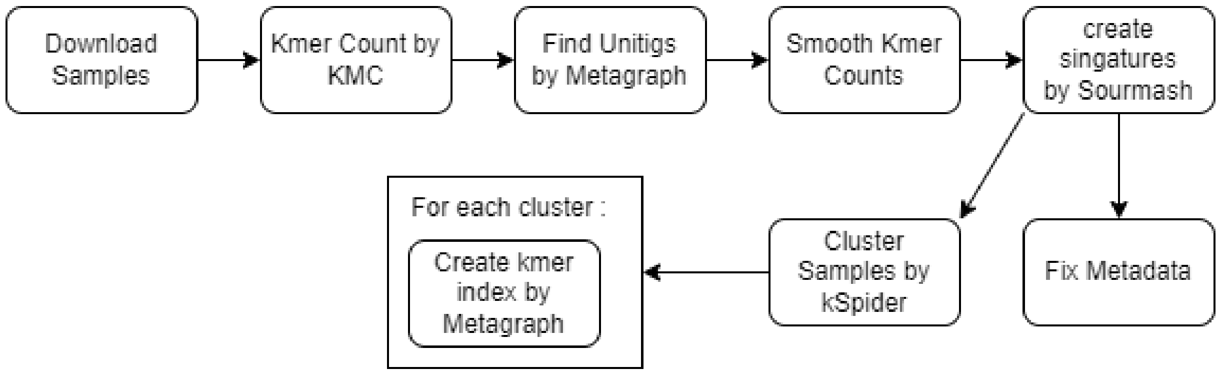
Download Samples
Kmer Count by KMC
Find Unitigs by Metagraph
Smooth Kmer Counts
create singatures by Sourmash
For each cluster :
Create kmer index by Metagraph
Cluster Samples by kSpider
Fix Metadata

Figure 3.1. **Indexing Workflow** The diagram outlines the workflow implemented to construct the kmer indexes for the thousands SRS samples. The process initiates with the downloading and preprocessing of the samples, involving the identification of unitigs and normalization of their counts. Sourmash is then employed to create signatures from the unitigs, facilitating swift comparison between samples. Any discrepancies in metadata are identified through the examination of Sourmash signatures. Subsequently, kSpider is utilized to develop a dendrogram for all the samples, and sample clusters are computed based on this dendrogram. Finally, a kmer index is generated for each cluster using metagraph.

Upon the download of each sample, we employ kmc for kmer counting, setting the minimum count to 3 to filter out singletons and doubletons, which are most likely errors. Metagraph is utilized to identify the unitigs and retain only the average kmer count per unitig, thus smoothing kmer counts. This smoothing reduces the size of the kmer counts to one-tenth while maintaining high genotyping accuracy.

From each unitig, we calculated sourmash signatures using a k size of 51 and subsampling to the scale of 1 to 10k, which entails keeping a hash for every 10,000 kmers. Sourmash signatures are very small and can be compared with each other very efficiently to gain insights about the datasets. We use the signatures to find discrepancies in the metadata. Moreover, they can be used to identify clusters in the samples to aid in creating the CCDG, instead of relying solely on the metadata for clustering. Therefore, errors in the metadata can be discovered by examining the clusters.

We next identify subsets of samples characterized by maximal intra-subset similarity. Initially, kSpider calculates pairwise similarities between all samples based on their Sourmash signatures. We employ hierarchical clustering using the Scipy library [81] to construct a dendrogram visualized in Figure 3.7 by iTOL [82]. Finally, we utilize Metagraph to construct CCDGs for each cluster independently.

### 3.3.2 The Great Genotyper

The Great Genotyper begins its workflow with a list of CCDGs and a list of variants, which can be either phased or unphased. As demonstrated in the green workflow in Figure 3.2, it processes all input variants across the samples in the CCDGs. The initial step entails extracting k-mers unique to the variant region. These k-mers are then used to query the counts in all samples present in the CCDGs. Unphased variants are genotyped by comparing the counts of the unique k-mers to the average sample coverage in Fast Genotype mode.

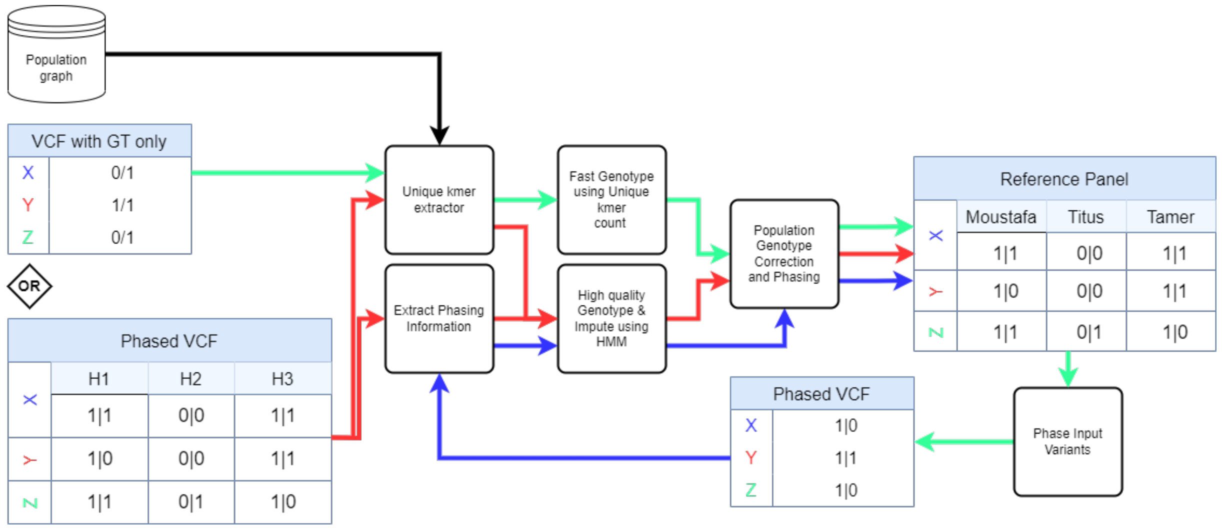

Figure 3.2. **The Great Genotyper workflows** The Figure describes three distinct workflows to create reference panels, each illustrated with a different color of arrow.: The red workflow creates a high-quality reference panel from phased variants, the green workflow creates the panel from variants without phasing information, and the blue workflow enhances the green panel by phasing the input variants then following the red workflow. Each of these workflows utilizes three specific processes: Unique k-mer Extractor, Extract Phasing Information, and High-Quality Genotype. These procedures are based on a scaled-up version of the Pangenie model, which enables the simultaneous processing of thousands of samples. "Population Genotype correction and phasing" scrutinize genotypes by evaluating the genotyping quality across all samples. Once completed, the removed genotypes are re-estimated using a statistical imputation process, implemented by Beagle [83, 80]. Beagle also uses the results from the population genotype to phase all variants, thereby generating a reference panel based on the input variants. Lastly, "Fast genotyping" produces initial genotypes for all the samples in the population database by comparing the k-mer counts of unique k-mers to the average sample coverage.

The genotype of unphased variants are identified using emission probability equations(3). Essentially, we compute the probability for all three possible genotypes and select the variant's genotype based on the highest probability, as described in Equation (3).

$$GT = \arg \max_{gt \in \{0/0, 0/1, 1/1\}} P(gt|KC) \tag{3}$$

In contrast, for phased variants, genotyping is achieved via a Hidden Markov Model (HMM) based on the Li-Stephen model [79], as implemented in Pangenie (the red workflow in Figure 3.2). This HMM fulfills a dual role of genotyping and imputation. Genotyping is done in the same way as the previous methods, and if the evidence for certain variants is low, the HMM will impute their genotypes provided it can genotype nearby variants on the same haplotype.

Following the primary genotyping of samples by either model, the genotyping results undergo a population level assessment and correction. The population-wide genotyping confidence for each variant allows the identification of the likely erroneous genotypes. Subsequently, those genotypes with low genotyping confidence undergo statistical imputation using the high confidence genotypes of the whole population as a reference panel. This step involves phasing of all the population variants as well.

As the HMM genotyping model demonstrates superior recall compared to the simple genotyping model for unphased variants, The Great Genotyper provides an optional second run to enhance the recall. In this second run, input variants are phased using the reference panel from the first run, and the HMM is employed, as indicated by the blue workflow in Figure 3.2.

### 3.3.3 Population Genotype correction and phasing

The preceding steps process each sample in isolation. In this step, we optimize the genotype quality of each sample by leveraging the power of all samples in the kmer index. The process initiates by identifying high-confidence genotypes in the initial results, produced by the sample-wise genotyping methods, either Fast or HMM. Following this, a reference panel is assembled, comprised of the high-confidence genotypes.

Subsequently, Beagle [83, 80] is employed to statistically impute the filtered, lowconfidence genotypes derived from the reference panel, simultaneously phasing the resultant variants, thereby yielding phased genotypes for all samples.

It is crucial to note that Beagle employs a different HMM model, albeit one very similar to the one used in the High-Quality mode. However, there is a divergence in the way linkage disequilibrium is calculated. In Beagle, linkage disequilibrium is computed statistically from the high-confidence genotypes within the created reference panel. In contrast, The High-Quality HMM step calculates it based on the phasing information provided by the user in the input variants.

The synergy between these two imputation methods does not only enhance the results of genotyping but also broadens the application scope for the Higher Quality HMM model, enabling its use even when phasing information is absent in the input VCF, as described in the blue arrows in figure 3.2.

The genotyping models yield a confidence measure for the output genotypes, deducing the likelihoods of all possible genotypes and selecting the one with the highest probability. The confidence level is derived from the difference between the highest probabilities and the other probabilities: a larger difference correlates to higher confidence.

The components driving these probabilities can primarily be distilled into two factors: the number of unique kmers discovered for each variant haplotype and the count of these kmers in the sample. The first factor is a constant across all samples since it is determined only from the reference genome and variant. However, the second factor varies per sample. Some samples may present robust evidence for a particular genotype, while others may not due to either low coverage of the region in the sample or the exhibition of a different haplotype not present in the input haplotypes.

To filter out low-confidence variants, we calculate the median of genotype confidences for each genotype and discard genotypes falling below this median. This approach allows us to establish a variable threshold calculated using the results from all the samples, providing a balanced way to sift through the variants. For variants abundant in unique kmers, this threshold will be high, while more challenging variants will have a lower threshold, accommodating the varying levels of confidence in different scenarios.

### 3.3.4 Experimental Design

This section outlines the experimental designs for the benchmarking experiments conducted to compare the accuracy of The Great Genotyper with the state-of-the-art genotyping tools: Pangenie, GraphTyper2, Paragraph, and GATK.

### 3.3.5 Genotyping Accuracy Experiment

The Genotyping accuracy experiment assesses the proficiency of genotyping tools in genotyping both small and structural variants. These variants are identified from the NA12878 sample and are then evaluated within short-read samples from HG00731, HG00512, and HG01891.

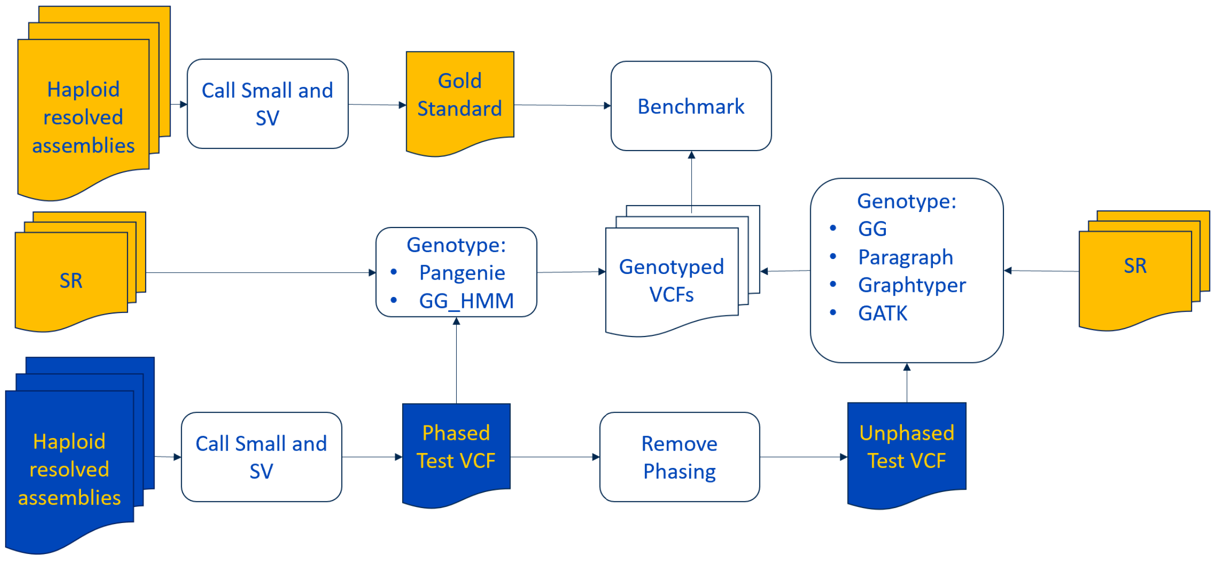

Figure 3.3. **Benchmark Genotyping Accuracy Workflow** The figure represents the workflow for the benchmarking experiment. The gold color represents samples: HG00731, HG00512, and HG01891, and the blue represents the NA12878 sample.

This experiment is structured into two components. The first component involves creating a Truth Set and Test Sets. A Truth Set is established using the same algorithm applied in Pangenie for creating phased variants (including small variants and structural variants) from haplotype-resolved assemblies. For HG00731 and HG00512, haplotype-resolved assemblies from [84] were utilized, while assemblies from the Human Pangenome

Project [85] were used for HG01891. This workflow aligns the assemblies to the reference genome using minimap2 [7] and identifies variants using PAV tools [84], applying each haplotype assembly independently. Subsequently, the resulting VCFs are merged using bcftools [86]. Additionally, the workflow generates a confidence region on the reference genome, ensuring that only one segment from the assembly maps to it to avert misassemblies. Variants within this confidence region are deemed suitable to represent the full truth in these regions.

For the test VCF, assemblies of NA12878 were downloaded from GIAB [42], and the identical workflow was applied to call the test variants. The overlap between the test variants and the Truth Sets is then calculated, with variants found in both samples being categorized as true positives and those found only in the test set as true negatives.

The second component is running the Genotypers and Benchmarking them. Subsequently, different genotypers are executed on the test variants (NA12878) and short-read samples of HG00731, HG00512, and HG01891. The genotyping results are then compared against the truth set. The benchmarking outcomes are stratified based on whether the variant is located in a repeat region or not and are also classified by type and size: SNP, Indel($< 50bp$), Insertions/deletions( $> 50bp$), and complex Insertions/Deletions, where "complex" denotes that the variation generates more than one breakpoint.

## 3.4 Results

### 3.4.1 Selecting Samples to Represent Human Genomic Diversity

To capture the genomic diversity of humanity, we compiled a sample list amalgamating three prominent sequencing projects: the 1000 Genome Project (1kGP) [87], the Human Genome Diversity Project (HGDP) [88], and the Simon Genome Diversity Project (SGDP) [89]. Each project harbors a distinctive vision and utilizes different resources for selecting samples to be sequenced, but all have conducted whole genome sequencing with an approximate depth of 30x.

The 1kGP has sequenced 3,202 samples encompassing 26 populations. Conversely, while HGDP and SGDP covered a more extensive range of populations, 56 and 137 respectively, they have fewer samples per population, totaling 828 and 276 respectively. The broader coverage of HGDP and SGDP enables the identification of common genomic variations across humanity. Additionally, the depth of the 1kGP provides a deeper understanding of genomic variations within selected populations.

This amalgamation yielded 4,271 unique samples and 35 redundant samples, representing 140 populations. The distribution of these unique samples is depicted in Figure 3.4. As illustrated in Figure 3.5, over 1,000 samples originate from Africa. East Asia, South Asia, and West Eurasia contribute around 800 samples each, with the remaining samples distributed across the Americas, the Middle East, and Oceania.

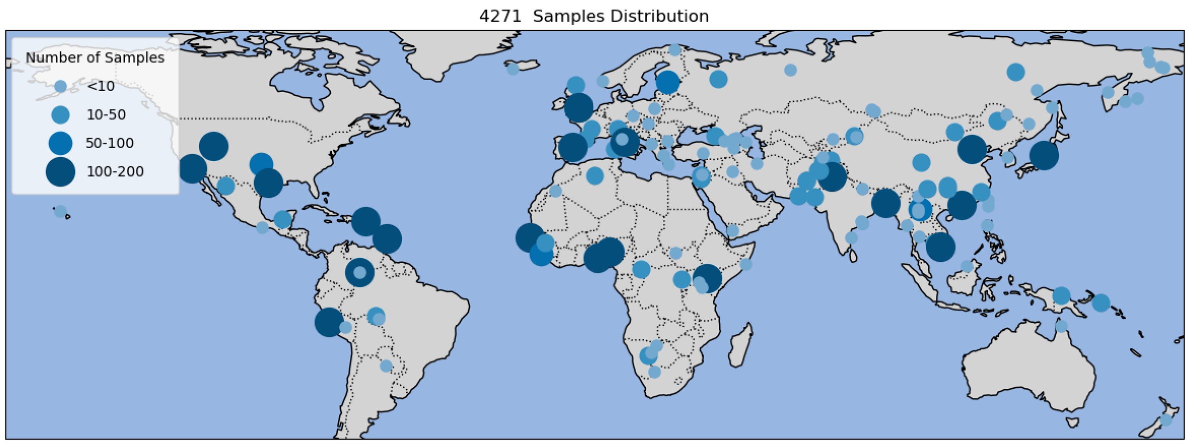

Figure 3.4. **Global Distribution of Samples** The map demonstrates the representational breadth of selected samples across global populations. Small circles denote samples from the SGDP and HGDP datasets, while the larger circles represent those from the 1KG project, providing an overview of the coverage of world populations by these samples.

### 3.4.2 Using Sourmash to Correct the Sex Fields of the Samples in the Metadata

We exploited the lightweight nature of Sourmash to perform fast comparisons over the signatures for quality control purposes. The sex of the samples was determined by calculating the containment between the sample's signature and the signature of chromosome Y from GRCh38 reference genome.

Male samples exhibited an average overlap ratio of 0.65(SD=0.039), while the average

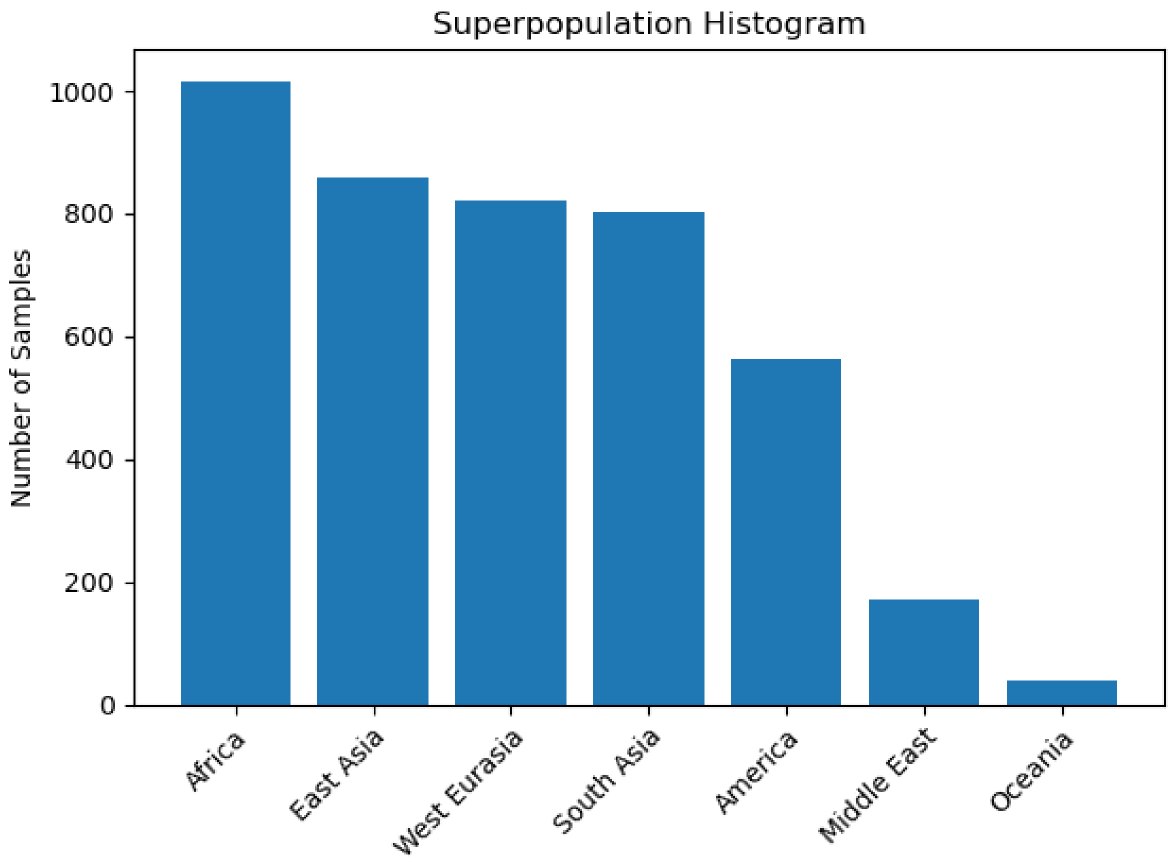

Figure 3.5. **Histogram of Samples Across Super Populations**:The histogram displays the distribution of samples from each superpopulation selected for indexing in the CCDG.

for female samples was 0.08(SD=0.003). We concluded that this method can reliably determine the sex of the samples.

We observed four discrepancies between the sex values calculated by Sourmash and the sex provided in the metadata. For further verification, we calculated the average coverage of reads on chrY from the cram files.

Results of the average coverage and Sourmash overlap ratio are illustrated in Figure 3.6. Both graphs exhibit the same pattern, leading us to conclude that the metadata needs adjustment.

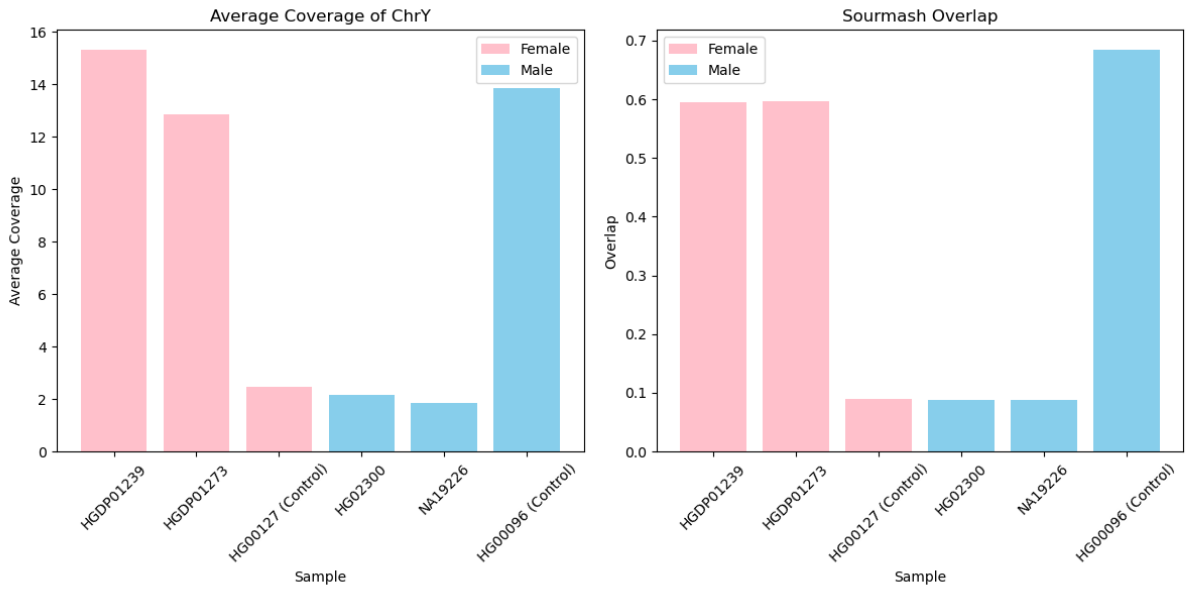

Figure 3.6. **Validating Metadata Accuracy**: This figure illustrates the discrepancies in metadata through a dual comparison. On one the left, it presents the average coverage of chrY for samples with incorrect metadata. On the right, it displays the overlap ratio between the sourmash signature of the samples and that of chrY. The color of the bars represents the sex as provided in the metadata, with blue bars denoting male and pink bars denoting female. Ideally, blue bars should be larger than pink bars; however, inconsistencies arise due to errors in the metadata. The ratio of control samples aligns with expectations.

### 3.4.3 Using kSpider to cluster the 4271 Samples into 28 clusters

Initially, kSpider calculated pairwise similarities between all samples based on their Sourmash signatures. We employed hierarchical clustering using the Scipy library [81] to construct a dendrogram visualized in Figure 3.7 by iTOL [82].

The initial attempt to construct a dendrogram was not entirely successful, with Figure 3.7.A illustrating clusters encompassing samples from various continents. Upon closer inspection, it was revealed that samples of identical sex but differing subpopulations were more similar to each other than to those of the opposite sex and same subpopulation. To address this sex bias, we subtracted the signature of chrY from all samples, resulting in a more homogeneous dendrogram, as depicted in Figure 3.7.B.

From the dendrogram, fifteen clusters were extracted and subsequently refined manually to ensure each encompasses between 100-350 samples, the number of samples per cluster is detailed in Figure 3.8. To illustrate the coherence of our clusters, they are plotted

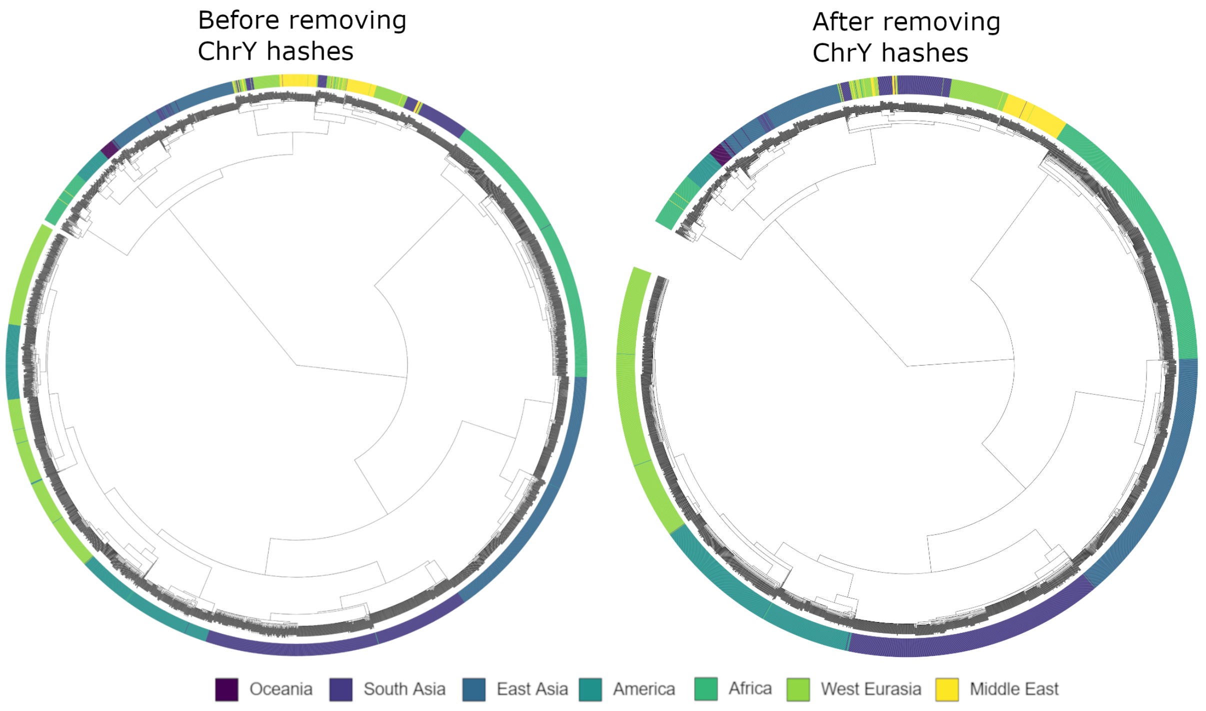

Figure 3.7. **Addressing chrY Bias in Clustering** The figure depicts two attempts to create a dendrogram for the 4271 samples. The outer circles consist of fine lines, each representing a sample, with the color of the lines signifying the population of the sample as per the metadata. The dendrogram is displayed within these circles, delineating the clusters. Excluding chrY hashes yields more homogeneous clusters, as illustrated in the left dendrogram.

on the world map in Figure 3.9, demonstrating how geographically proximate populations tend to be grouped into the same cluster. Additionally, thirteen samples that did not align with any clusters were prudently omitted to mitigate the risk of contamination.

### 3.4.4 Effect of Error Cleaning and Counts smoothing on genotyping Accuracy and the CCDG size

We investigated the influence of Metagraph's sample preprocessing on genotyping accuracy to determine the best parameters for optimal results. The error cleaning and kmer count smoothing algorithms employed by Metagraph are outlined in subsection 3.2.1. We generated several CCDGs from subsamples of the HG00731 SRS at coverages of 5x, 10x, 20x, and 30x. Each CCDG was constructed using different parameters, which are summarized in Table 3.1, along with the final sizes of the CCDGs. Subsequently, we assessed the precision and recall of genotyping by conducting the experiment described in Subsection

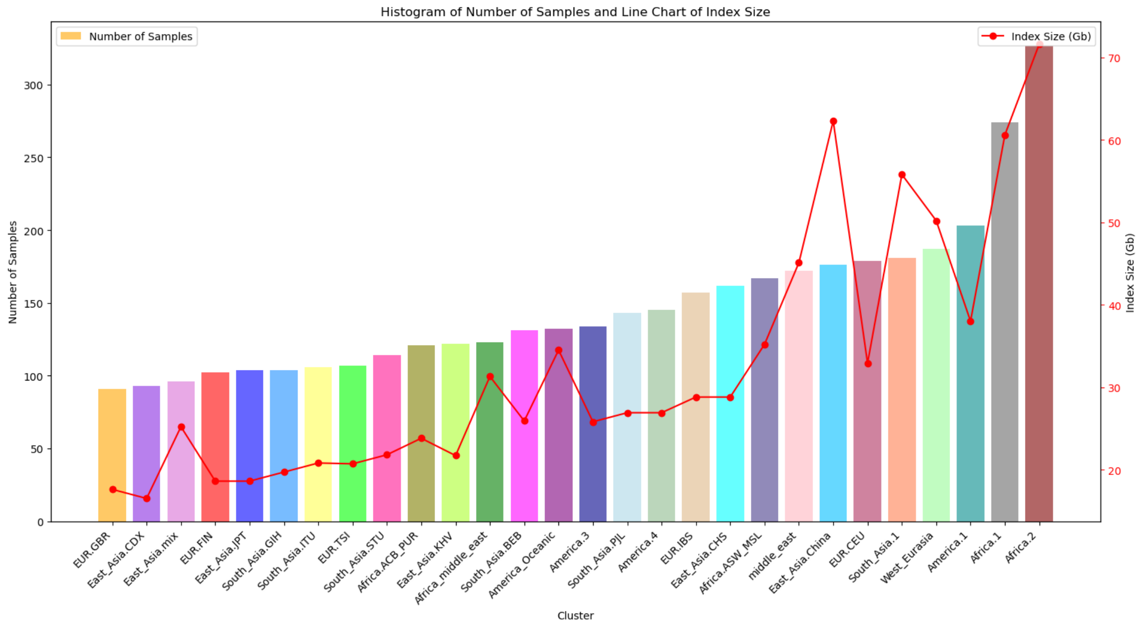

Figure 3.8. **Number of Samples per Cluster and index size** This figure depicts the distribution of samples across each cluster, as determined by the clustering algorithm, on the left y-axis. Concurrently, on the right y-axis, it showcases the corresponding final index size for each cluster.

3.3.5.

Additionally, we considered logging the kmer counts as a preprocessing step. While our benchmarking methodology aligned with the approach described in subsection 3.3.5, for this study, we created the input CCDG from subsamples of HG00731 SRS at coverages of 5x, 10x, 20x, and 30x. The resultant CCDGs sizes is showcased in table

Results in figure 3.10 and table 3.1 indicate that preprocessing methods do not impact samples with coverage exceeding 20x. For coverages of 10x and 5x, logging the counts was the most influential, significantly decreasing both the f-score and the final CCDG size. On the other hand, smoothing led to a nominal drop in the f-score but notably reduced the CCDG size. Cleaning had a moderate impact on the f-score and caused a slight reduction in the CCDG size. The findings from this experiment were instrumental in guiding our decision-making regarding preprocessing parameters.

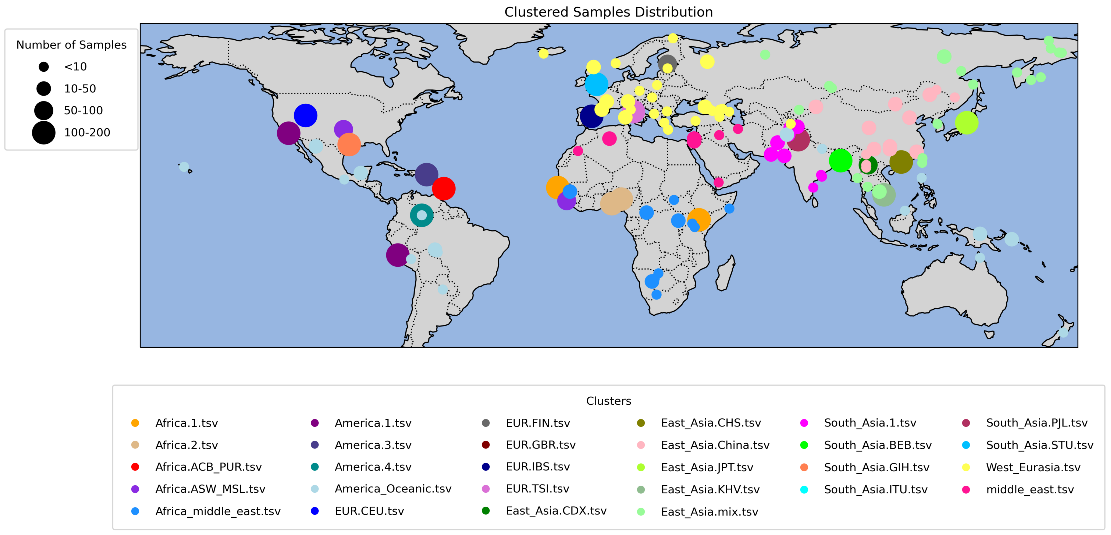

Figure 3.9. **Visualizing Clusters on a World Map** We are visualizing the outcome of a clustering algorithm on a world map. Each circle on the map represents a population. The size of each circle shows the number of samples we have from that population, and the color of each circle shows the cluster to which most samples from that population are assigned. The visualization of the clustering results generally aligns well with the geographical locations on the map, with some exceptions. This is because the coordinates on the map represent where the samples were analyzed, not the original locations of the populations. For instance, we have African samples (colored red), European samples (colored blue), and Gujarati Indian samples (colored light blue) that were sequenced in the US, and Sri Lankan samples sequenced in the UK.

### 3.4.5 Using Metagraph to Create 920 GB CCDGs from 95 TB of CRAM Files

We employed the Metagraph indexing workflow to generate a distinct index for every cluster. From an initial size of 95 TB in the input CRAM files, the total size of all clusters was reduced to 925.2 GB, yielding a space savings of 100x. Figure 3.8 graphically presents the size of each cluster in tandem with the count of samples they contain

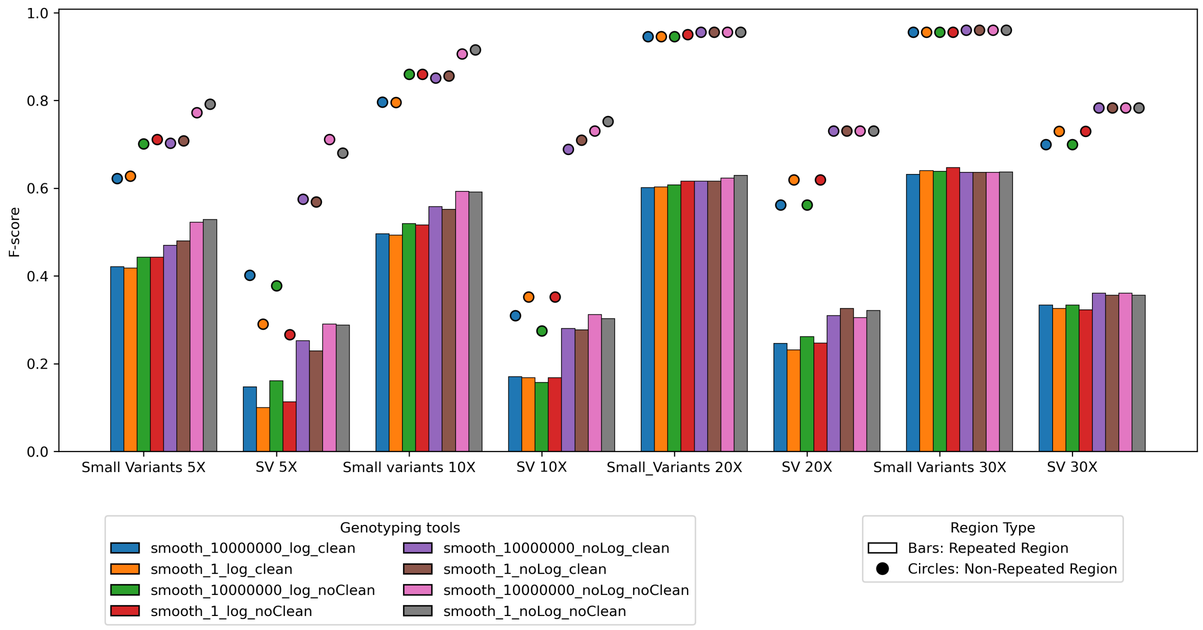

Figure 3.10. **F-score comparison between the different preprocessing parameters** Metagraph's preprocessing techniques influence the genotyping f-score. These techniques include *Smoothing Counts*, where only the average kmer count per unitig for each sample(smooth10000000) is retained instead of preserving the individual kmer counts(smooth1); *Log Counts*, which involves saving the log of kmer counts to conserve space; and *Clean*, the error cleaning algorithm in the Metagraph framework described in subsection 3.2.1.

| Index Name | Smoothed Counts | Log Counts | Cleaned | CCDG size(GB) |
|---|---|---|---|---|
| smooth10000000_log_clean | Y | Y | Y | 1.42 |
| smooth10000000_log_noClean | Y | Y | N | 1.59 |
| smooth10000000_noLog_clean | Y | N | Y | 1.62 |
| smooth10000000_noLog_noClean | Y | N | N | 1.80 |
| smooth1_log_clean | N | Y | Y | 1.76 |
| smooth1_log_noClean | N | Y | N | 1.92 |
| smooth1_noLog_clean | N | N | Y | 3.00 |
| smooth1_noLog_noClean | N | N | N | 3.30 |

Table 3.1. **Impact of Metagraph Preprocessing on CCDG size:** The table demonstrates how preprocessing methods like Smoothing Counts, Log Counts, and the Clean algorithm alter the size of the final CCDG. Smoothing Counts simplifies kmer data to averages per unitig, Log Counts compresses kmer information logarithmic ally, and Clean removes errors, as detailed in Metagraph's framework described in subsection 3.2.1.

### 3.4.6 Achieving Population Genotyping on Standard HPC Servers in a Matter of Hours

The performance evaluation of the Great Genotyper was conducted in two modes: Fast-genotyping and High quality(HMM). Using the Fast genotyping mode, the Great Genotyper efficiently genotyped 700k variants in 1,000 samples within a span of just 72 minutes. In contrast, when operated in the High-quality mode, the processing time increased to 3 hours and 45 minutes. Notably, both modes effectively utilized a single machine without exceeding a memory usage of 150 GB, as delineated in Figure 3.11.

To contextualize this efficiency, Pangenie required close to an hour to genotype the 700k variants in one sample, while GraphTyper2 consumed approximately two hours for the same task. If these durations are extrapolated to accommodate the processing of 1,000 30x samples, both Pangenie and GraphTyper2 would presumably necessitate several weeks to complete the genotyping process using a single machine. This underscores the remarkable proficiency of the Great Genotyper in both its operational modes.

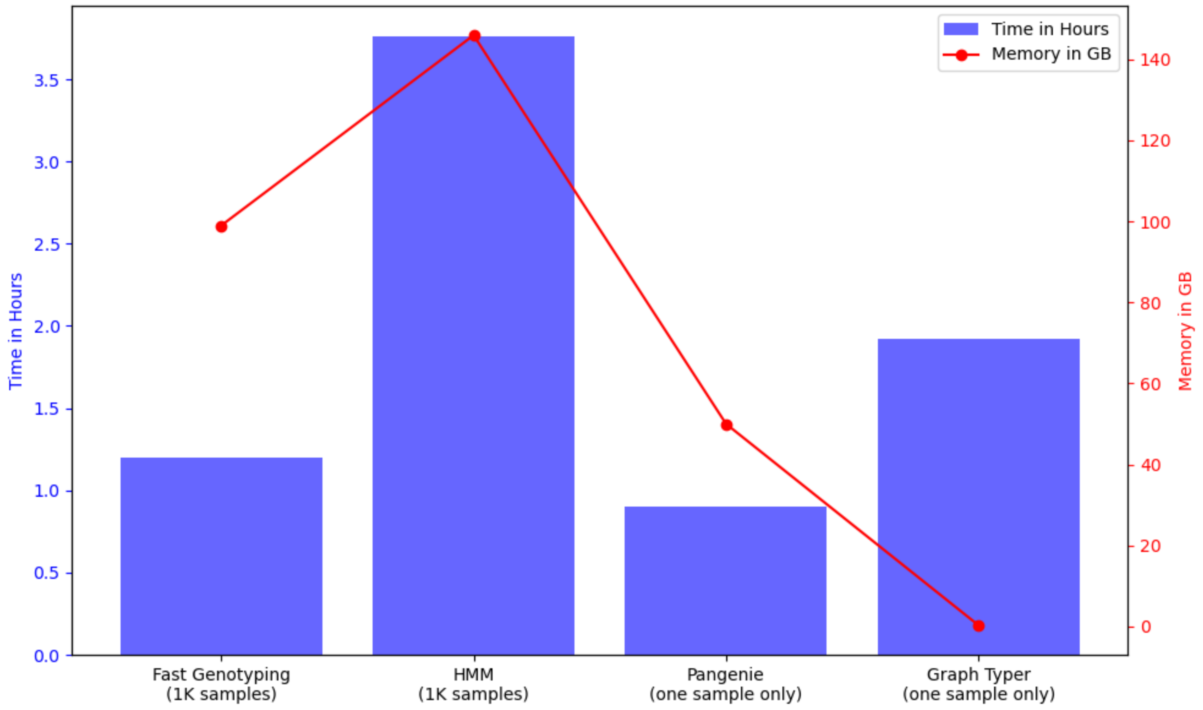

Figure 3.11. **Performance comparison between The Great Genotyper and the state of the art** Running time and memory of different tools used to genotype 700K phased variants (SV and small variants). The Great Genotyper is genotyping 1,000 30x samples, while the other genotypers are working on only one 30x sample.

### 3.4.7 Computational Efficiency has Almost no Cost on Accuracy

We executed the experiment detailed in subsection 3.3.5 to compare the precision and recall of the Great Genotyper with other state-of-the-art genotypers. In this experiment, the genotypers were provided with SV and small variants derived from the NA12878 haploid-resolved assemblies, alongside a short read sample from 30x SRS of HG00731. Additionally, The Great Genotyper was supplied with an extra 1000 samples to harness population information, thereby refining the genotyping accuracy. The results can be seen in Figures 3.12. The Great Genotyper's HMM and Pangenie demonstrates a superior f-score for SV variants in most scenarios. When focusing on small variants, it maintains a comparable f-score to GATK, except in the context of genotyping complex indels. Fast Genotyping generally has a lower F-score than other methods, but a second pass significantly improves the f-score. It surpasses the performance of other methods in many cases, but still ranks just behind the Great Genotyper's HMM. In summary, the Great Genotyper consistently delivers competitive or superior genotyping accuracy across different scenarios.

Figure 3.13 shows the effect of sample coverage on the F-score of The Great Genotyper and other methods. All methods showed an expected decrease in accuracy for coverages 5x and 10x. However, Pangenie, The Great Genotyper HMM, are giving the best results for genotyping SV at low coverage. The Great Genotyper Kmers-only genotyping model is affected by lowering the sample's coverage, but the F-score was significantly enhanced using the second pass HMM.

### 3.4.8 Assessing the Phasing Accuracy of The Great Genotyper

In Figure 3.12, we observe a significant improvement in the genotyping f-score of unphased variants using two second pass methods. This enhancement is achieved by initially phasing the unphased input variants based on their genotyping results within the population via fast genotyping. Subsequently, this phased information is harnessed by the second pass using the high-quality HMM model. To evaluate the quality of phasing, we carried out an experiment, as shown in Figure 3.14. It reveals that the switch error rates fluctuate between 4-7%.

For this experiment, we began with phased small variants and SVs sourced from haploid-resolved assemblies representing diverse subpopulations. We then stripped these variants of their phasing information and introduced the unphased variants to the fast

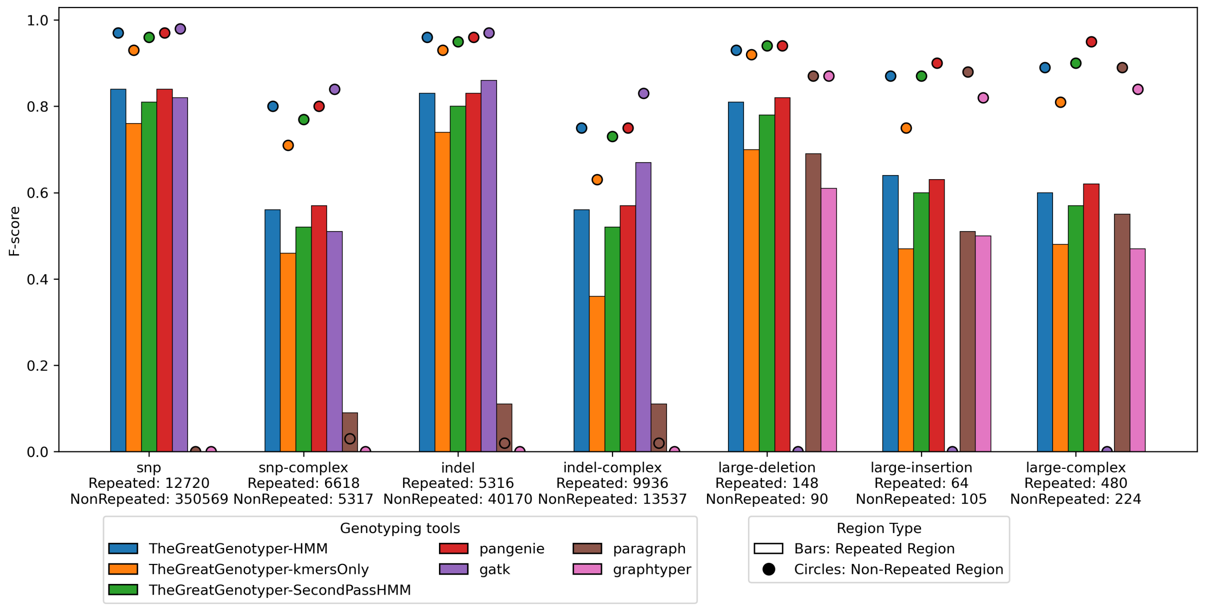

Figure 3.12. **F-score comparison between The Great Genotyper and the state of the art** The figure illustrates the F-scores of different genotyping methods, differentiating between SNPs/indels (under 50 bp) and larger insertions/deletions (above 50 bp). F-scores for Variants located in repeated regions are shown as bars, while those in non-repeated regions are shown as circles. Additionally, Variants are categorized based on the complexity of their genomic location.

genotyping model within The Great Genotyper, genotyping them across 1000 samples. The outcome from this stage then serves as a reference panel to phase the unphased variants. Finally, we compared the results against the original phased variants using whatshap [90].

## 3.5 Use Cases

We have investigated how the unparalleled efficiency and accuracy of The Great Genotyper at the population level can revolutionize genomic studies. In this section, we utilize a subset of the index from 1000 samples to demonstrate various use cases. Firstly, we examined The Great Genotyper's potential in Mendelian disease studies by genotyping the entire ClinVar database and comparing the allele frequencies of pathogenic and benign variants. Additionally, we employed The Great Genotyper to genotype HPRC pangenome variants across the 1000 samples. Our findings indicate that expanding high-quality variants from the pangenome to a larger number of samples paves the way for applications

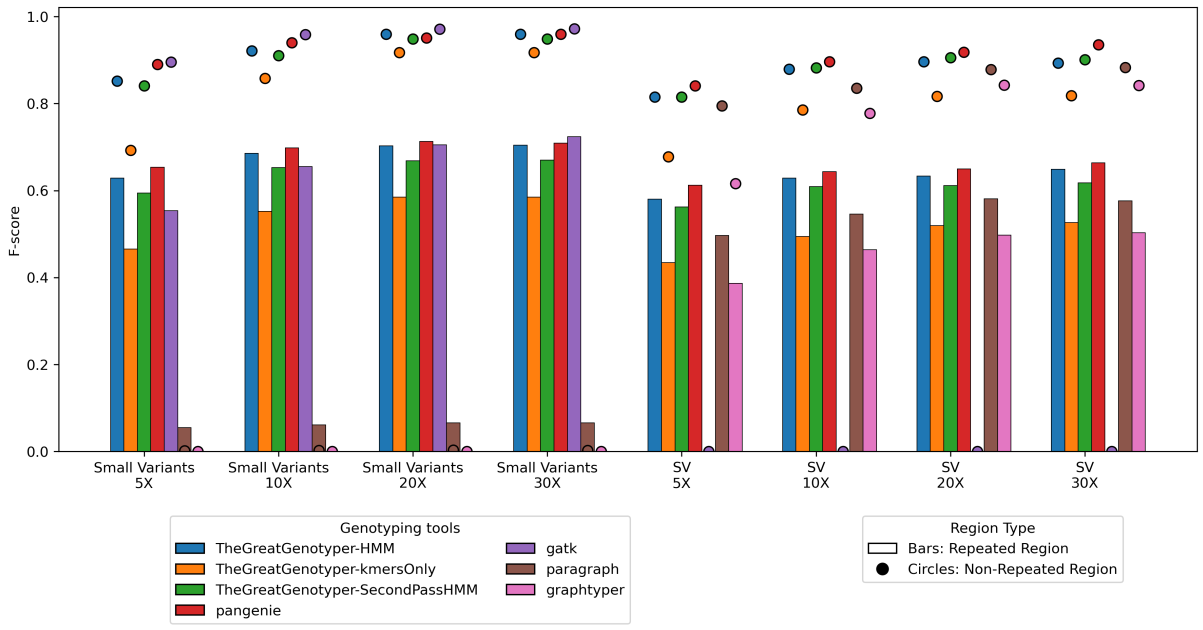

Figure 3.13. **Coverage effect on genotyping F-score.** The figure illustrates the effect of coverage on the F-scores of different genotyping methods, differentiating between small variants (under 50 bp) and SV (above 50 bp). F-scores for variants located in repeated regions are shown as bars, while those in non-repeated regions are shown as circles. Additionally, variants are categorized based on the complexity of their genomic location.

such as phasing and imputation of small as well as structural variants in unprecedentedly large cohorts. Furthermore, it provides insights into the haplotype structure of these variants, enabling the fine mapping of GWAS SNPs using SVs from the pangenome.

### 3.5.1 The Great Genotyper can help finding pathogenic variants

We used the fast genotyping method to genotype the ClinVar database variants in the 1000 samples. In Figure 3.15, we visualized the allele frequency results for both benign and pathogenic variants using box plots, stratified by their significance. Almost all pathogenic variants had zero allele frequency in the healthy population, while benign variants showed a wider range of frequencies. This indicates that calculating allele frequencies of candidate variants in indices of 1000s sample can be a reliable metric for prioritizing rare variants when investigating their pathogenicity

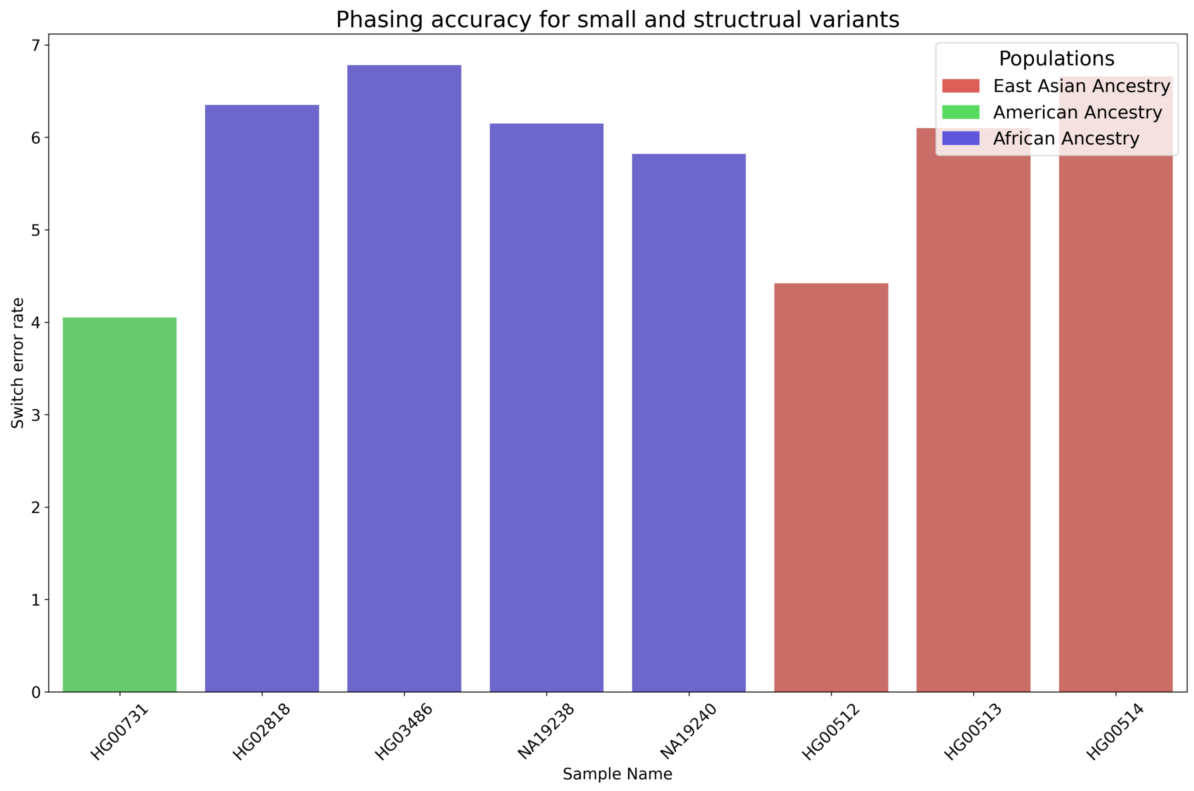

Figure 3.14. **Phasing Accuracy of The Great Genotyper Across Ethnically Diverse Samples** The figure illustrates the phasing accuracy of The Great Genotyper, showcasing switch error rates derived by comparing its phasing results against haplotype-resolved assemblies. The analysis encompasses 8 samples from a variety of ethnic backgrounds.

### 3.5.2 Expanding the Pangenome: Genotyping HPRC Variants in 1000 Samples with The Great Genotype

We downloaded the HPRC pangenome in the form of a decomposed VCF to ensure compatibility with the Great Genotyper. The VCFs contained 8.5 M variants, and the counts per variant type are summarized in Table 3.2. Subsequently, we genotyped these variants across three chromosomes: chr1, chr5, and chr20, within the 1000 samples present in the prebuilt CCDG. The resulting output maps HPRC variants to GRCh38 coordinates and provides phased genotypes for these samples. To evaluate the representation of the HPRC pangenome within our dataset, we analyzed the allele frequencies across the human population. Figure 3.16 displays a histogram of these frequencies, stratified by population. This visualization clearly indicates a balanced representation of variants

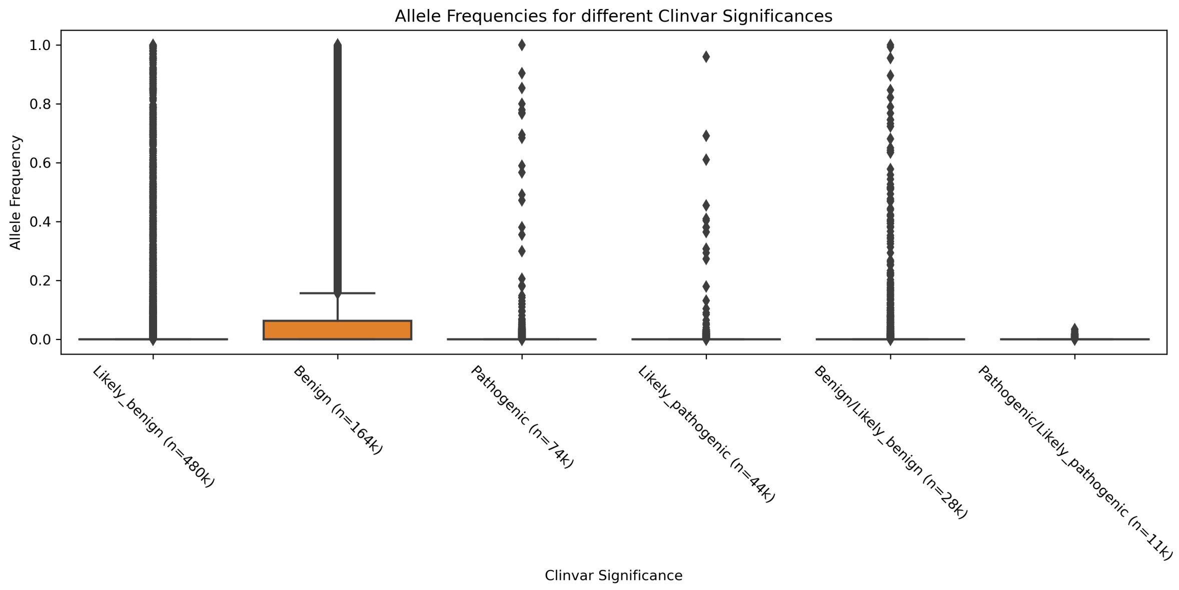

Figure 3.15. **Allele Frequency distribution for variants in Clinvar** The box plots summarize the distribution of allele frequencies for variants in the ClinVar database, categorized by their clinical significance.

across most populations.

| Variant Type | Count |
|---|---|
| SNV | 3,807,964 |
| Indels(<50 bp): | 1,346,326 |
| Deletions | 597,069 |
| Insertions | 65,7889 |
| Complex | 91,368 |
| SV(>50bp): | 84,445 |
| Deletions | 17,188 |
| Insertion | 48,988 |
| Complex | 18,269 |

Table 3.2. **Variant Counts in HPRC Pangenome** The table details the number of variants from the decomposed VCF of the HPRC pangenome for Chromosomes chr1, chr5, and chr20, categorized by their type.

The generated VCF is termed as the 'extended pangenome'. We computed the Principal Component Analysis (PCA) for the extended pangenome using the Hail library. The

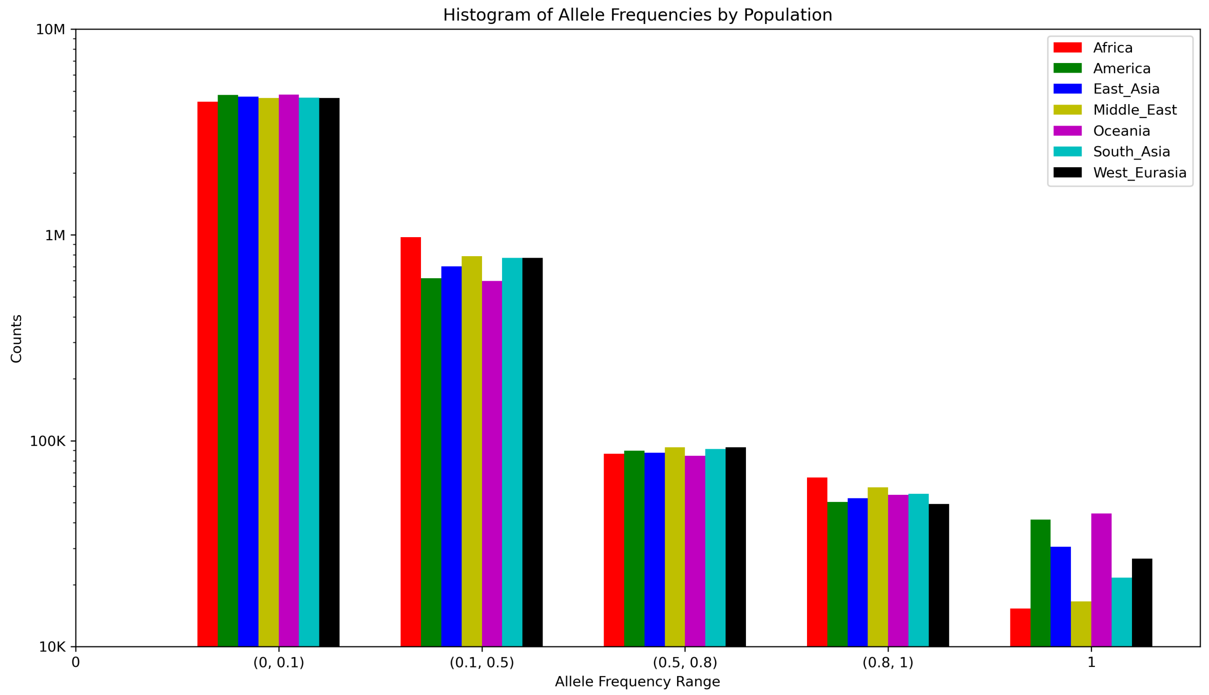

Figure 3.16. **Distribution of HPRC Allele Frequency in human populations.** This histogram depicts the number of HPRC pangenome variants within distinct allele frequency ranges stratified by population.

resulting V-shaped visualization is presented in Figure 3.17. In the sections that follow, we'll discuss how leveraging these high-quality variants can enable a variety of genomic applications.

## 3.5.3 Impute SV by using the extended pangenome

The extended pangenome can serve as a reference panel akin to reference panels produced by the 1kGP project [87] but with SV for the first time. In this section, we demonstrate the precision and recall of imputing both small and structural variants.

Our methodology follows an experiment detailed in GLnexus [91]. Pseudo-microarray call variants were generated for Genome in a Bottle (GIAB) [42] benchmark samples (HG002, HG003, HG005, and HG006). We extracted variants from the truth sets at sites utilized in Illumina Infinium OmniExpress-24, simulating the genotyping of those samples with this microarray.

Subsequently, we used the extended pangenome to impute the small and structural variants. We employed the 1kG reference panel [87] exclusively for the imputation of small

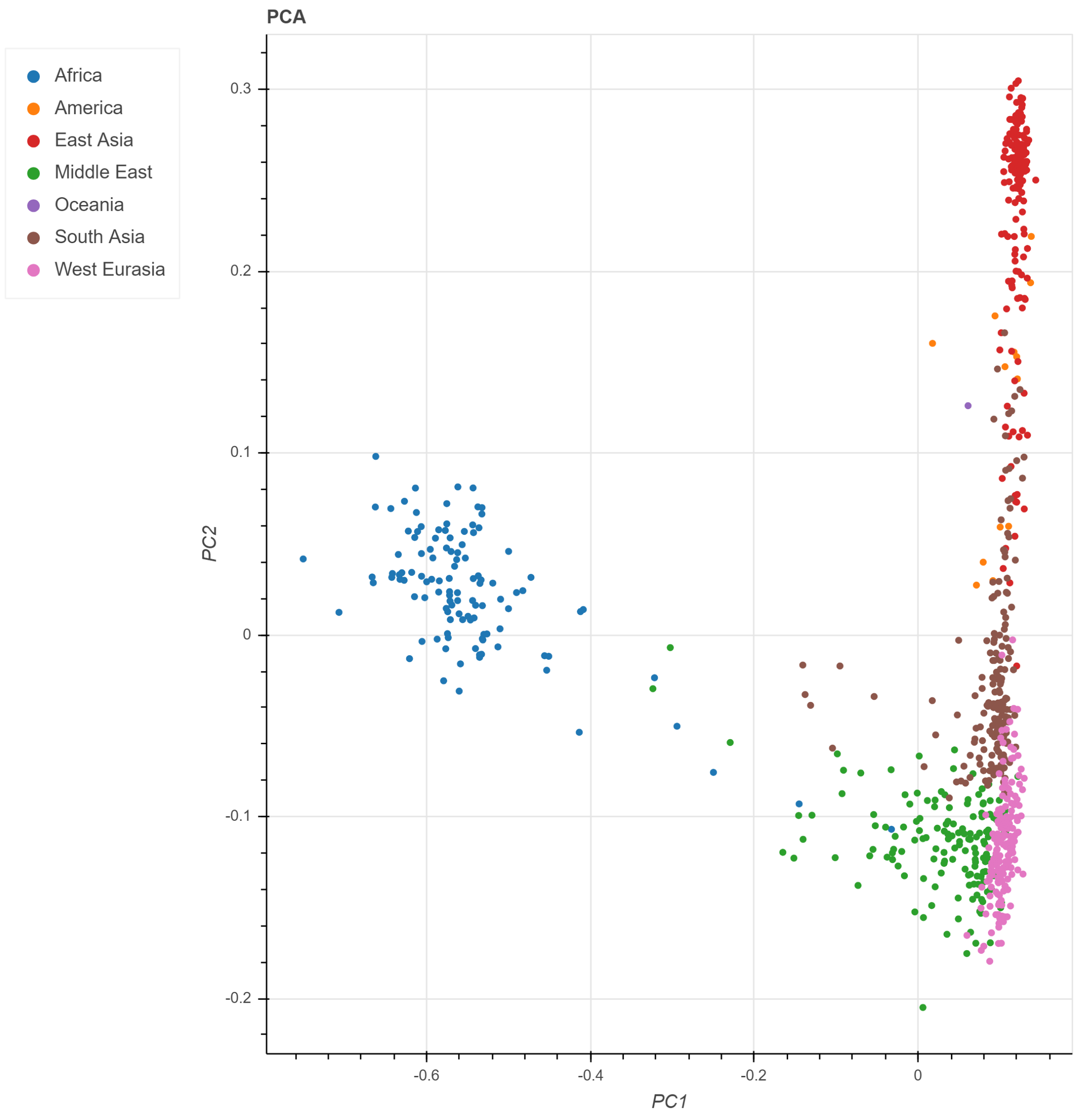

Figure 3.17. **Principal Component Analysis (PCA) of genetic data from chr20 of the extended pangenome.** Each dot represents an individual, with colors indicating their respective population groups. The clustering based on the genetic data aligns well with the individuals' geographical origins.

variants. For small variant imputation, we conducted benchmarking using hap.py(v0.3.12) against the GIAB truth sets. Structural variant imputation for sample HG002 was benchmarked against NIST(V0.6) [92] with truvari [93] (v3.5.0).

The extended pangenome exhibited commendable precision and recall for small variant imputation, as depicted in figure 3.18. When compared to the 1kG reference panel, it displayed marginally reduced precision. In terms of recall, the extended pangenome exhibited decreased recall for SNPs but a significantly higher recall for indel imputation. These results hint at the potential of enhancing imputation panels by capitalizing on the high-quality variants of the pangenome.

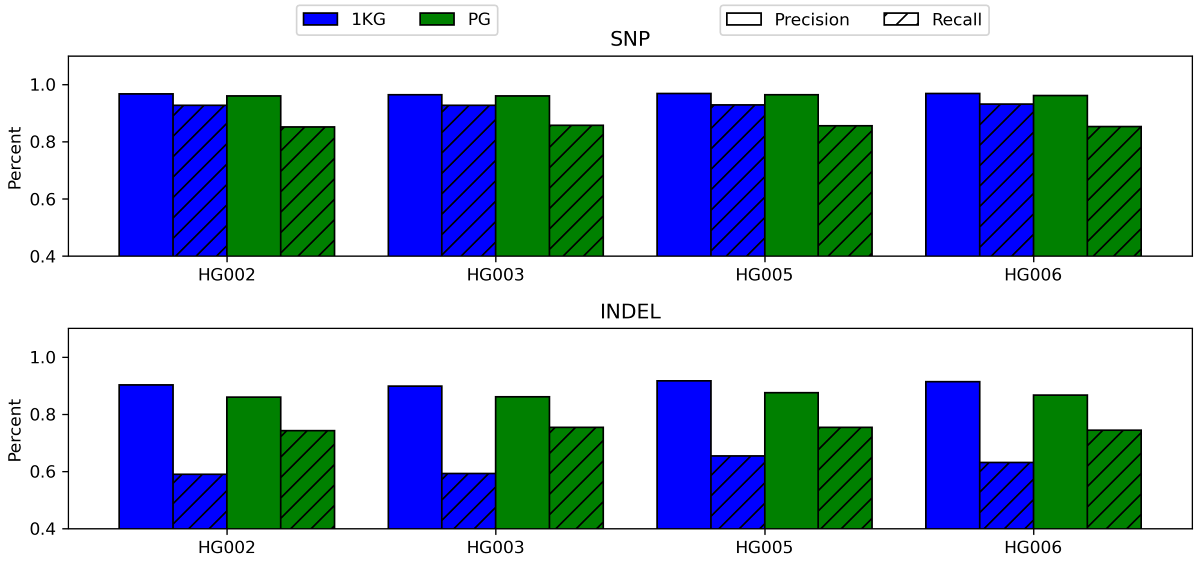

Figure 3.18. **Imputation Accuracy Comparison: 1kGP vs. Pangenome Panels.** This figure presents a comparison of imputation precision and recall between using the 1kGP reference panel (in blue) and the pangenome reference panel (in green).

Conversely, the extended pangenome showcased remarkable results in imputing common structural variants, achieving a precision and recall of 65%, as illustrated in figure 3.19. This imputation recall surpasses the recall of SV calling from 30x SRS using Manta. However, employing SNPs and indels called from 30x SRS via DeepVariant as input for SV imputation resulted in both lower precision and recall than imputation using the pseudo microarray. In conclusion, extended pangenomes can also be leveraged to augment microarray genotypes with common SVs.

### 3.5.4 Fine Map GWAS snps using SVs from the extended Human Pangenome

The extended pangenome provides detailed insights into the haplotype structure of common variants. To elucidate the potential phenotypic impacts of these variants, we explored the Linkage Disequilibrium (LD) between SVs and neighboring variants. Our aim was

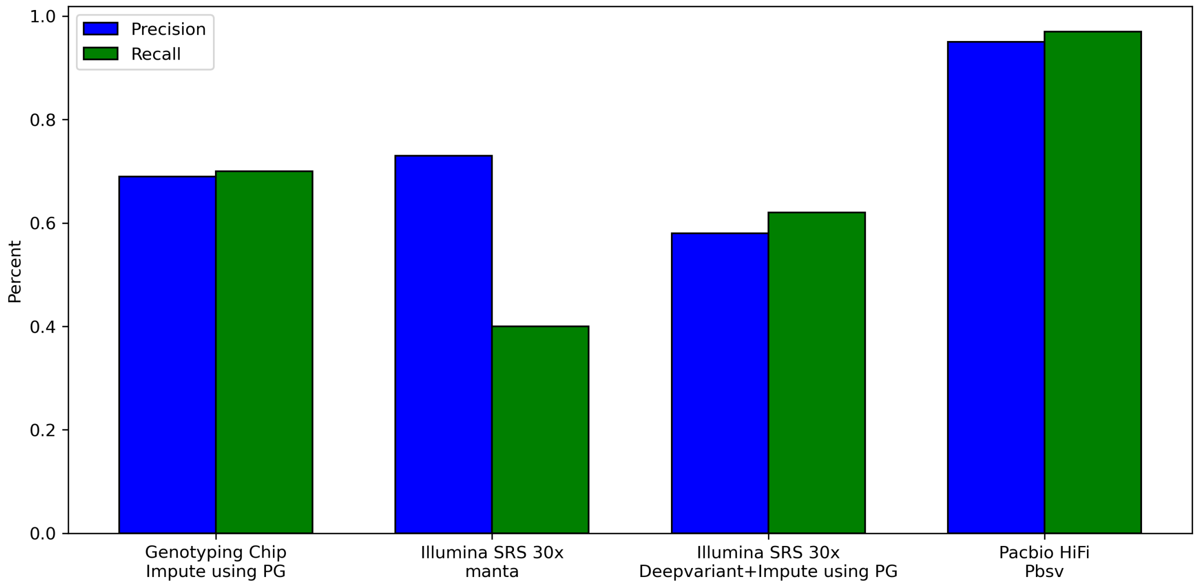

Figure 3.19. **SV Imputation vs. Direct Calling Accuracy** The figure contrasts the precision and recall of structural variant (SV) imputation using an extended pangenome against the direct variant calling from short-read data with Manta and from PacBio HiFi data using PBSV.

to identify co-occurring variants that might offer insights into their phenotypic effects. We initiated our investigation by annotating the SVs in chromosomes 1, 5, and 20 using AnnotSV [94] (v3.3.6). This revealed that approximately 4,000 SVs affect gene structures, as illustrated in table 3.3. Proceeding further, we computed the pairwise LD for each of these 4,000 variants with all the variants located within a 1MB window surrounding them. Our analysis indicated that 2,012 SVs exhibited a strong association with a neighboring variant, having an $r^2$ value greater than 0.8.

We utilized the identified associations to illuminate potential causal variants in GWAS studies. Searching the GWAS catalog, we found that 253 of the 4k SVs have a strong association with a GWAS SNP. Notably, 12 of these SVs affect the exons of genes. This list can be instrumental in elucidating the phenotypic effects of the SVs and pinpointing the causal variants for the traits examined in the GWAS study.

We delved deeper into one particular example from our list. We identified GWAS SNP rs12567976, which is linked to the 'High light scatter reticulocyte percentage of red cells', to be in linkage disequilibrium with two SVs. The SNP rs12567976 exhibits

| **SV Affecting Gene Structure** | |
|---|---|
| Affecting Introns | 3866 |
| Affecting Exons | 169 |
| Total | 4035 |
| **SV Affecting Genes & in LD with other variants** | |
| Affecting Introns | 1941 |
| Affecting Exons | 71 |
| Total | 2012 |
| **SV Affecting Genes & in LD with GWAS SNPs** | |
| Affecting Introns | 241 |
| Affecting Exons | 12 |
| Total | 253 |

Table 3.3. **Distribution of SVs affecting gene function and in LD with GWAS SNPs.** This table summarizes the count of SVs that affect gene structures and have association with GWAS SNPs or other variants.

a strong association ($r^2 = 0.9903$) with the variant (ID=>5549119>5549121), a 22-bp deletion at Chr1:212897047-212897068, affecting an exon of FLVCR1. Furthermore, it has a strong association ($r^2 = 0.9865$) with another variant (ID =>5548884>5548886), a 32-bp deletion at Chr1:212891335-212891366, impacting an intron of the same gene. A visualization of the LD heatmap can be seen in figure 3.20. This figure highlights a distinct haplotype block spanning 1KB, providing insights into the genetic variance mechanisms of the FLVCR1 gene.

## 3.6 Discussion

The Great Genotyper serves as a practical solution for population genotyping at very large scales. It offers the capability to genotype a new set of variants, whether small or structural, in 4K SRS samples in just a matter of hours. More importantly, it provides a novel solution for the chronic N+l problem by eliminating the need to download a cumbersome 100 TB of CRAM files.

Instead, the Great Genotyper operates using a prebuilt CCDG for human populations

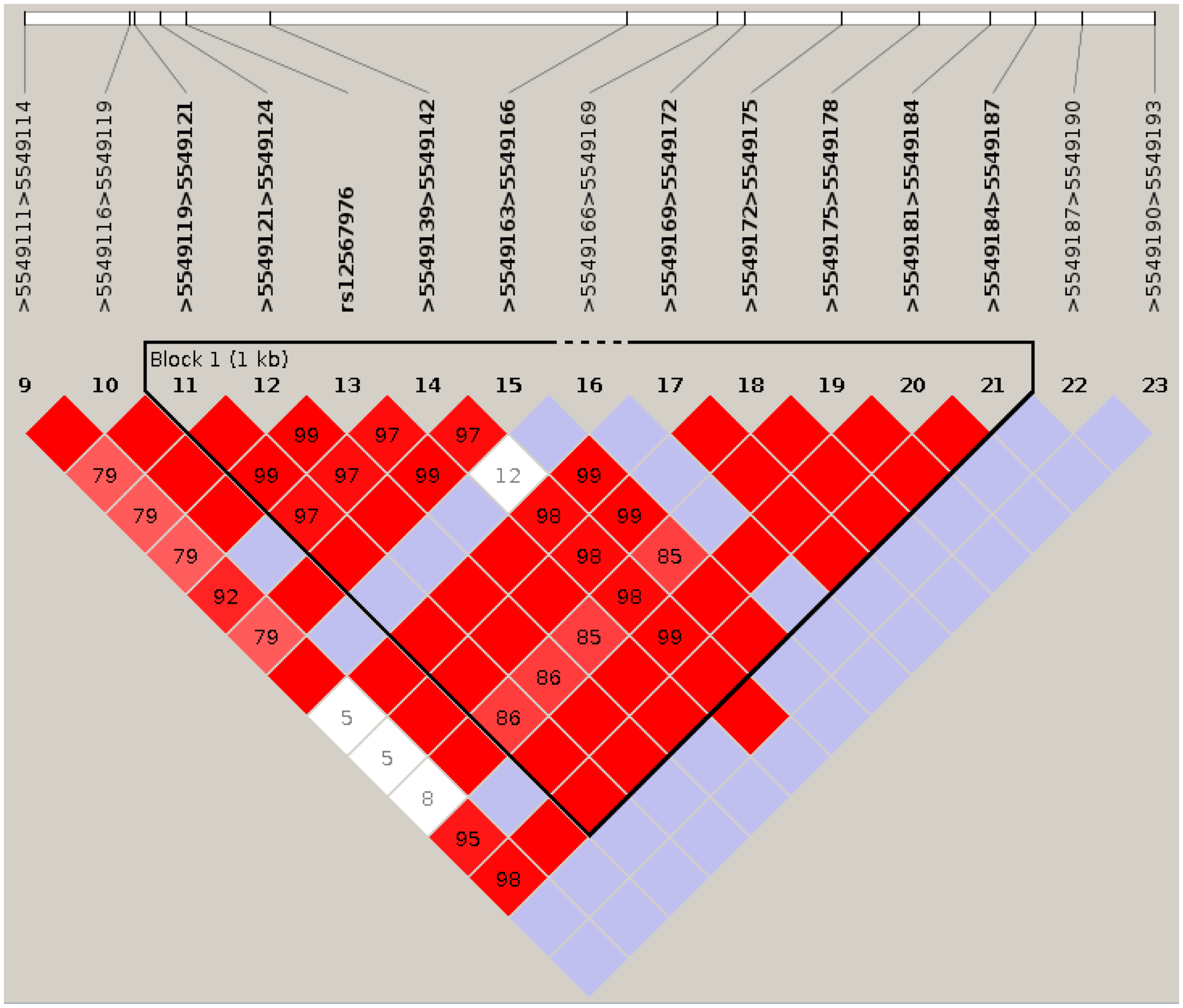

Figure 3.20. **Linkage Disequilibrium heatmap for haplotype block around rs12567976**: The figure depicts the linkage disequilibrium for rs12567976 in relation to 9 variants from the extended pangenome. Among them, the variant with ID **>5548884>5548886** represents a 32-bp deletion, and the variant with ID **>5549119>5549121** is a 22-bp deletion. Both of these deletions impact the gene structure of FLVCR1.

that is approximately 1 TB in size, effectively decoupling the intensive data preprocessing from the actual genotyping process. In terms of input, the Great Genotyper is versatile; it accepts any set of phased or unphased variants in VCF format, along with the reference genome. The outcome is the phased genotypes of all input variants in the indexed samples.

A significant advantage is that the Great Genotyper does not sacrifice quality for scalability. On the contrary, the scalability feature empowers the tool to jointly genotype thousands of samples, which, in turn, enhances the genotyping quality even more. kmer-based genotypers such as Nebula and Pangenie have previously demonstrated the potential of kmers for precise genotyping. They leveraged the specificity of certain kmers to variants, using shifts in the counts of these kmers as indicators to genotype the variants. The Great Genotyper elevates this approach, considering the counts of these kmers across an entire population of samples. This innovation facilitates the calculation of a confidence measure for each genotype based on the collective population data.

Furthermore, the tool is equipped to impute missed genotypes through a two-tiered approach. Initially, imputation is rooted in the phasing information of the variants, either provided as input or derived from the CCDG. Subsequently, the Great Genotyper integrates Beagle, leveraging the high-confidence genotypes within the population to further impute genotypes. This dual-phase imputation process ensures that the Great Genotyper can deliver performance on par with Pangenie, even if some data is compromised during the kmer count preprocessing prior to the creation of CCDG.

The enhanced accuracy and scalability of the Great Genotyper has paved the way for new applications in genomics. For instance, accurate allele frequencies can now be directly derived from sequences rather than merging information from sparse studies or variation databases that rely on variant calling in SRS studies. Accurate determination of allele frequencies plays a pivotal role in pinpointing causal variants in disease-gene discovery studies.

Furthermore, simultaneous genotyping and phasing of common variants enables unprecedented resolution in understanding the haplotype structure with and across populations. As an example, we genotyped variants of the HPRC pangenome in 1,000 samples to produce what we call an "Extended Pangenome (EP)". We presented how EP can be used to impute common structural variations (SVs) with a recall rate that surpasses some short reads caller like Manta.

Taking our analysis further, we explored the Extended Pangenome for SVs in high linkage disequilibrium with GWAS SNPs. We limited our focus to 4,000 SV variants impacting gene structures in three chromosomes. Intriguingly, we discovered that half of these SVs exhibited strong associations with at least one GWAS SNP. We are optimistic that our findings will contribute to a deeper comprehension of the relationship between genotype and phenotype concerning these structral variants.

The Great Genotyper has revolutionized scalable and accurate population genotyping by representing variants and SRS samples in kmer space. The tool achieves its scalability by preprocessing the SRS samples into CCDG, effectively decoupling the intensive data preprocessing from the actual genotyping process. This preprocessing also significantly reduces the data size from the SRS raw reads or CRAM format, facilitating easier sharing and reuse of the CCDG.

Although the Great Genotyper has proven effective in generating high-quality genotypes for both small and structural variants, it does have certain limitations. Firstly, some variants cannot produce specific kmers because the kmers from the alternate sequences may also exist in other parts of the genome. Such variants cannot be genotyped precisely by kmer-based approaches. This limitation, however, is partially offset through imputation. Furthermore, genotyping copy number variants is beyond the capabilities of the current version of the Great Genotyper. While it is not an insurmountable challenge, it would require that a dedicated genotyping model be developed. Another constraint is that the Great Genotyper utilizes two separate imputation models, as they are implemented in two distinct tools, Pangenie and Beagle. A unified model tailored specifically for imputing genotypes using the kmers in the CCDG might not only enhance the accuracy but could also boost the performance.

The Great Genotyper is the inaugural step in utilizing kmer indexing for population genotyping, opening many doors for future genomic applications. Creating more CCDGs to represent specific subpopulations or individuals exhibiting specific traits, like autism, is crucial for understanding the role of genomics in these cohorts. Moreover, while most population studies have been conducted on humans [95], it isn't due to a lack of data. The Sequence Read Archive (SRA) [96] is a vast reservoir of short-read samples for non-human organisms. Generating CCDGs for these samples could facilitate population-scale studies for other species.

The current CCDG for the human population, and the additional CCDGs to be created for other cohorts, are invaluable resources with potential applications that extend beyond genotyping. For instance, variants can be directly called from the graph using methods such as corticall [97]. Additionally, it can aid in subsetting pangenomes by selecting segments of the pangenome that have kmers present in a specific population, thereby creating a more streamlined pangenome tailored to that population. We encourage the community to explore and uncover more ways to harness the extensive genomic diversity revealed by the CCDG.

In conclusion, The Great Genotyper has transformed population genotyping into a routine task with its flexible CCDG representation of populations. Its scalability allows improving genotyping quality by using population information. This tool's practicality aids in expanding variant lists into broader dimensions, revealing complex genomic details. We demonstrated its potential in applications such as creating SV imputation panels, finding SV associations with variants from databases like the GWAS catalog, and accurately calculating allele frequencies. The CCDG, comprising 4,000 human samples, contains a vast genomic variation spectrum, accessible through The Great Genotyper or other methods, leading to enhanced genomic insights. Producing more CCDGs for additional cohorts or species will further optimize the use of existing SRS samples.

## 3.7 List of abbreviations

- SNV: Single Nucleotide Variant
- SV: Structural Variant
- SNP: Single Nucleotide Polymorphism
- SRS: Short Read Sequencing
- LRS: Long Read Sequencing
- HPRC: Human Pangenome Reference Consortium
- 1kGP: 1000 Genome Project
- HGDP: Human Genome Diversity Project

- SGDP: Simons Genome Diversity Project
- GIAB: Genome in a Bottle
- CCDG: Counting Colored Debruijn Graph
- HMM: Hidden Markov Model
- PCA: Principal Component Analysis
- LD: Linkage Disequilibrium
- EP: Extended Pangenome
- SRA: Sequence Read Archive

# Chapter 4

# Cattle Structural Variant Catalog

## 4.1 Introduction

Structural variants (SVs) are genomic alterations exceeding 50 base pairs in size and can manifest as insertions, deletions, inversions, copy number variations, or complex rearrangements. Such variants have been demonstrated in humans to disrupt genes and significantly affect gene expression [98]. Investigating SVs across different cattle breeds is essential for identifying functionally impactful variants that correlate with economically important traits. For instance, research has revealed associations between SVs in bovine genomes and key agricultural traits such as fertility [99], immunity [100], and milk production [101]. The creation of a comprehensive SV catalog encompassing a wide array of breeds is vital for advancing bovine genomic research, elucidating complex traits, and providing insights to livestock breeders.

Creating a catalog of common SVs to represent the vast diversity of cattle breeds presents significant challenges. It necessitates the collection of a large number of samples to uncover the genome diversity between the breeds and to compute statistics such as allele frequency, which help differentiate common variants from rare ones. Short Read Sequencing (SRS) technologies offer a cost-effective approach for detecting SVs across numerous samples. However, studies have indicated that SV calling from SRS data may suffer from unreliable precision and recall [6]. Conversely, Long Read Sequencing (LRS) has shown unprecedented performance in identifying SVs through mapping or assembly-based methods. Nevertheless, the cost of LRS remains prohibitive for extensive application across the many cattle breeds. Additionally, the efficacy of these methods has primarily been benchmarked using human genomes, which exhibit less genomic diversity compared to the variation found between cattle breeds. Further research is necessary to evaluate the performance of SV callers in the bovine context and to optimize their parameters.

Previous studies have developed SV catalogs for *Bos taurus* utilizing either SRS or LRS methodologies. Catalogs based on SRS have generally yielded a smaller number of SVs compared to those derived from LRS. For instance, Lee et al. [102] identified approximately 13K SVs by sequencing 266 Hereford samples with SRS, while Chen et al. [103] detected 17 SVs across 316 *Bos taurus* samples from Holstein and Jersey breeds. Conversely, Leonard et al. [104] uncovered 90K variants by generating a pangenome from three bovine trios using LRS. Beyond the realm of *Bos taurus*, comprehensive SV catalogs for *Bos indicus* were created using both SRS and LRS data. Dai et al. [20] compiled a significant catalog of 156,000 SVs. They used 20 partially phased genomes from 10 breeds, obtained through LRS. These variants were then genotyped using paragraph [66] in 402 SRS genomes, representing 76 breeds. In conclusion, SV catalogs for *Bos taurus* often lack either broad breed representation or the high quality SV discovery. A combination of both SRS and LRS is essential to address these deficiencies.

In our study, we developed a catalog of common SV for *Bos taurus* breeds. Our study began by assessing various SV calling methods, drawing on data from the Brahman and Angus trio assembly project [105]. Through this evaluation, we discovered that although SV calling from haplotype-resolved assemblies yields the highest precision and recall, SVs identified using LRS and SRS also contribute true variants not detected by the assembly-based approach.

Therefore, our strategy involved compiling an initial call set that integrated SV call sets of both SRS and LRS samples. These samples represented 16 breeds. Also, we created a Colored Counting De Bruijn Graph (CCDG) for 977 high-quality SRS samples covering 80 breeds. The Great Genotyper used the CCDG to genotype the massive initial callset in the 977 samples, and allele frequency was calculated from the results. We focused on

SVs with an allele frequency greater than 0.05 to create a catalog of common SVs. The filtered variants are assumed to be either rare variants or false positives.

The resulting catalog encompasses 175K common SVs. Upon performing functional annotation of these variants, we found that they have an impact on 15K genes and 26K transcripts, offering significant insights into the genetic architecture of *Bos taurus* breeds.

## 4.2 Methods

### 4.2.1 Evaluating SV Calling Techniques for Cattle Genomes with Different Technologies

We conducted an experiment to evaluate various techniques for calling SV on cattle genomes, as illustrated in Figure 4.1. We utilized a comprehensive dataset from the trio assembly of the F1 Cross of the Brahman and Angus project [105]. Within this project, the calf was sequenced using both Illumina and PacBio CLR reads, while both parents were sequenced using only Illumina. High-quality haplotype-resolved assemblies were generated for Brahman and Angus. We called the variants with respect to ARS-UCD1.2 [21] as follows:

- **Haplotype-resolved assemblies** were mapped using Minimap2 [7] and called the variants using SVIM-ASM [106].
- **PacBio CLR** were mapped using pbmm2 [107] and called the variants using both PBSV and CuteSV [11]. Then, SURVIVOR [108] was used to merge the variants.
- **Illumina SRS** were mapped using BWA [8] and called the variants with Manta [9] and Delly [10]. Variants were merged with SURVIVOR as well.

To evaluate the results of the callers, we employed two methods. Firstly, we deemed variants discovered by two different methods (SRS, LRS, or Haplotype-resolved assemblies) as true. Secondly, we genotyped the results from all the callers into fifty animals using Paragraph [66] and considered variants found in three or more animals as true. Although the second method might miss rare true variants and cannot be used as the sole benchmark method, it serves to corroborate the findings of the first evaluation method as an independent source. The results of this experiment are detailed in Section 4.3.1.

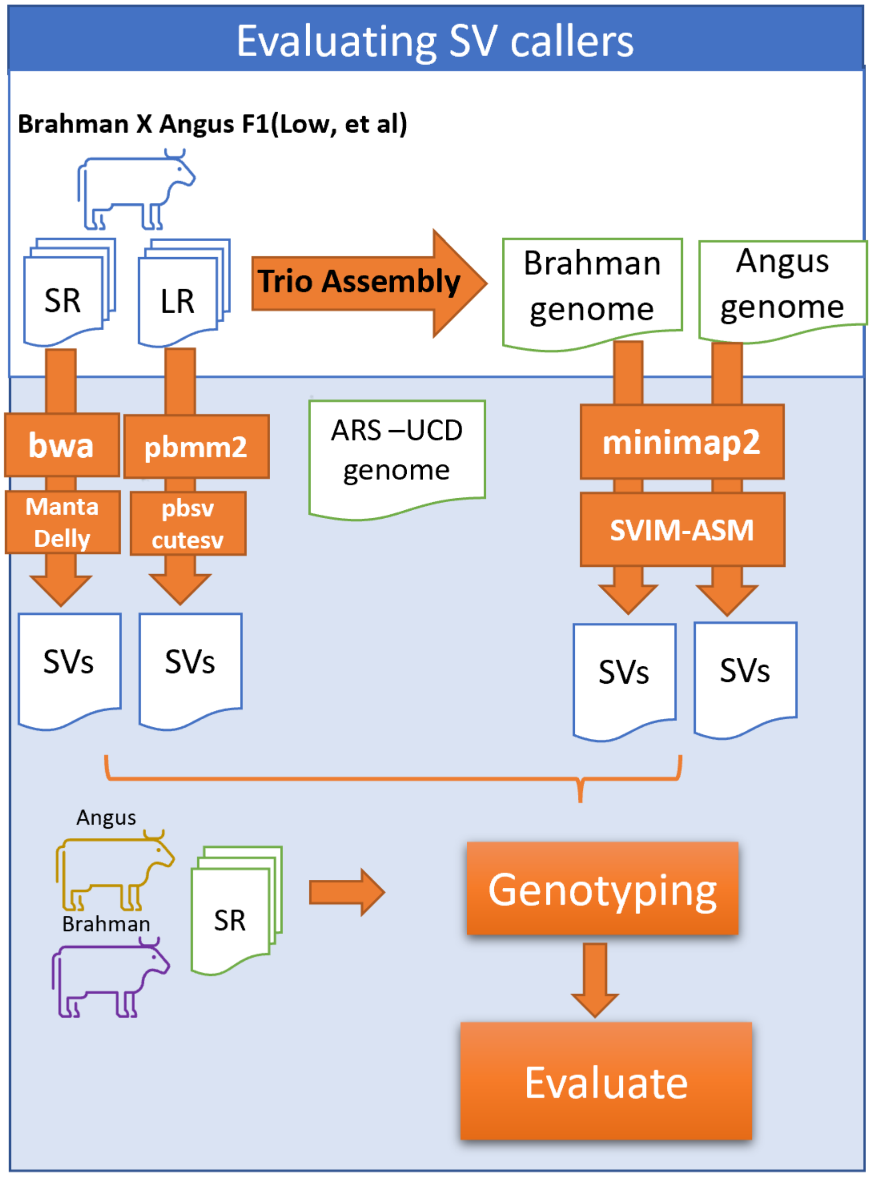

Figure 4.1. **Evaluation Workflow of SV Callers for Cattle Genomes** The white rectangle at the top of the figure describes the trio assembly project for the Angus and Brahman F1 cross [105]. We utilized the short reads, long reads, and haploid-resolved assemblies from this project to call structural variants. All discovered samples were genotyped in 50 short-read samples. The performance of each SV caller was then evaluated based on the identified SVs and the confirmation through genotyping.

### 4.2.2 Workflow for Creating a Catalog of Common Structural Variants in *Bos taurus* Breeds

The workflow for creating the common SV catalog involved calling variants from various sources to ensure the comprehensive discovery of possible variants, as depicted in Figure 4.2 using data described in Table 4.2.2. We employed bwa(v0.7.8), pbmm2(v1.12), and minimap2(v2.26) for mapping SRS, LRS, and assemblies respectively. For SV calling from SRS, manta(v1.6.0) was utilized. A combination of three tools—pbsv(v2.6.0), cuteSV(v2.0.3), and sniffles(v2.2)—was used for calling SV from LRS. Variant calling from assemblies was accomplished using SVIM-ASM(v1.0.3) for haploid non-resolved assembly genomes and Phased Assembly Variant Caller (PAV)(v1.0.3) for haploid resolved assemblies, such as trio assemblies.

| **Source** | **Source Type** | **NCBI BioProject Accession** | **Breeds** |
|---|---|---|---|
| Trio Assemblies | SRS, LRS, Trio Assemblies | PRJEB42335 | Nellore, Brown Swiss, Gaur, Piedmontese, and Original Braunvieh |
| Brahman & Angus trio assembly | SRS, LRS, Trio Assemblies | PRJNA603764 | Brahman and Angus |
| GCA_947034695.1 | Assembly | PRJEB55064 | CHAROLAIS |
| GCA_028973685.2 | Assembly | PRJNA927262 | Hanwoo |
| GCA_021234555.1 | Assembly | PRJNA667556 | Jersey |
| GCA_905123885.1 | Assembly | PRJEB41564 | Ankole |
| GCA_905123515.1 | Assembly | PRJEB41519 | N'Dama |
| GCA_021347905.1 | Assembly | PRJNA668863 | Holstein-Friesian |
| GCA_024542955.1 | Assembly | PRJNA849594 | Arequipa |
| GCA_000003205.6 | Assembly | PRJNA12555 | Brown Swiss |
| GCF_000003205.7 | Assembly | PRJNA12555 | Brown Swiss |
| GCF_000003055.6 | Assembly | PRJNA32899 | Hereford |
| Ankole | SRS, LRS | PRJEB39282 | Ankole |
| Indicus SV catalog | SV catalog | | *Bos indicus* |
| Ensemble SNV catalog | SNV catalog | | |

Table 4.1. **Catalog Construction Sample Sources:** This table enumerates the origins of samples utilized to construct the catalog, detailing their NCBI BioProject Accessions and corresponding breeds. The Indicus SV catalog was sourced from Indicuis Pangenome [20] , while the Ensemble SNV catalog was acquired from the Ensemble website.

The workflow employed The Great Genotyper to determine the allele frequency of all discovered variants in a population of 977 *Bos taurus* samples. False positives and rare variants were excluded from the initial call set by filtering out variants with an allele frequency (AF) less than 0.05. The filtered variants were then merged using Jasmine to produce the catalog for SVs.

Lastly, the workflow utilized Beagle to impute genotypes that were missed by The Great Genotyper due to insufficient coverage and to phase the variants as well. To aid Beagle in discovering haplotypes and improving imputation quality, the SVs were augmented with SNVs from the Ensembl catalog (V110). Subsequently, the output from Beagle was used to calculate the allele frequencies of the SVs in the catalog.

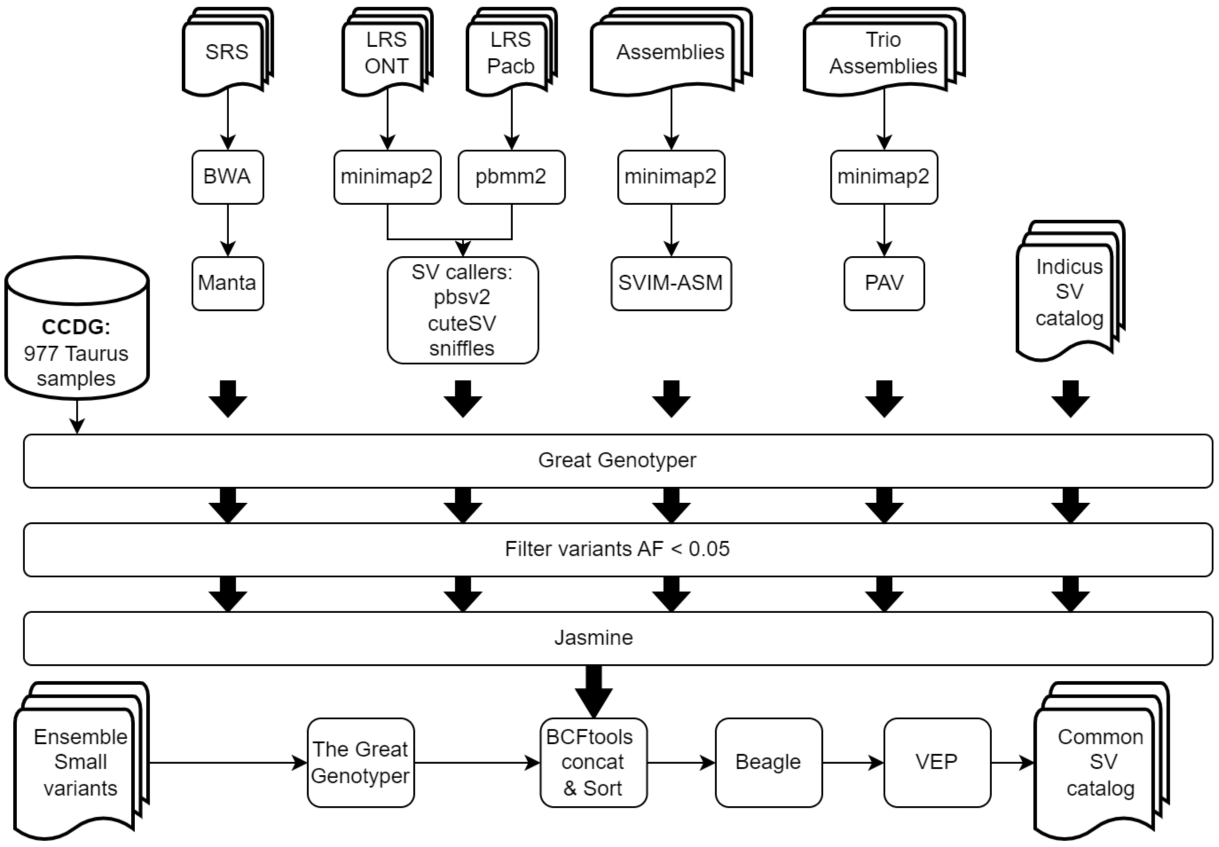

Figure 4.2. **Workflow diagram for creating a catalog of common structural variants for *Bos taurus* breeds**

## 4.3 Results

### 4.3.1 Use all the data: Trio assemblies, long and short reads

We evaluated the performance of SV callers on cattle genomes, as detailed in Section 4.2.1. The results of this experiment are presented in an upset plot, shown in Figure 4.3. Bars representing the trio assembly consistently exhibit high AF support. This holds true even for variants identified in the trio assembly but not recognized by any other caller. On the other hand, while variants identified by short reads alone do demonstrate a higher false positive rate, a considerable fraction, especially those larger than 10K in size, have high AF support. Importantly, variants identified by both short and long reads consistently show strong population support which can be a strong indicator of misassemblies. Understanding the performance of each technique proved invaluable in designing our workflow for creating a comprehensive SV catalog for cattle breeds.

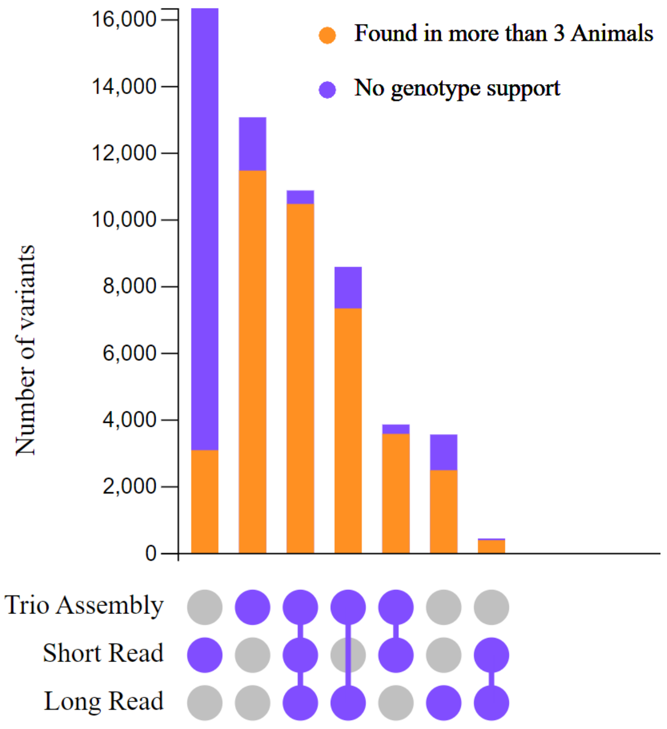

Figure 4.3. **Upset Plot Comparing different SV callers** The upset plot shows the intersection between the output of different callers. The orange part of the bars shows the percentage of those variants can be genotyped in at least three samples.

### 4.3.2 Building a CCDG Representing Different Cattle Breeds

A CCDG of 1,103 samples was constructed to represent various cattle breeds. This enables The Great Genotyper to utilize it, facilitating the genotyping of variants within the cattle population and ensuring accurate allele frequency calculations. The pipeline started by downloading 1,752 Samples from the SRA, with their associated metadata. Using KMC [77], we counted the kmers for all samples and subsequently determined the average genome coverage from the kmer counts histogram. After that, unitigs of each sample were constructed by metagraph [75]. Lastly, we used sourmash [109] to calculate kmer signatures at the scale of 10k.

Some samples from the SRA might be contaminated or exhibit excessive sequencing errors. To filter out potentially erroneous samples, we relied on the average coverage and the size of the sourmash signature. Samples with a coverage of less than 3 were excluded, as The Great Genotyper would struggle to differentiate between genuine heterozygous kmers and sequencing errors. Additionally, the scatter plot in Figure 4.4 reveals that the majority of samples possess a sourmash signature size ranging from 150K to 210K scaled hashes. Using both these criteria, we filtered out 649 samples. Both the chosen and filtered samples are illustrated in Figure 4.4.

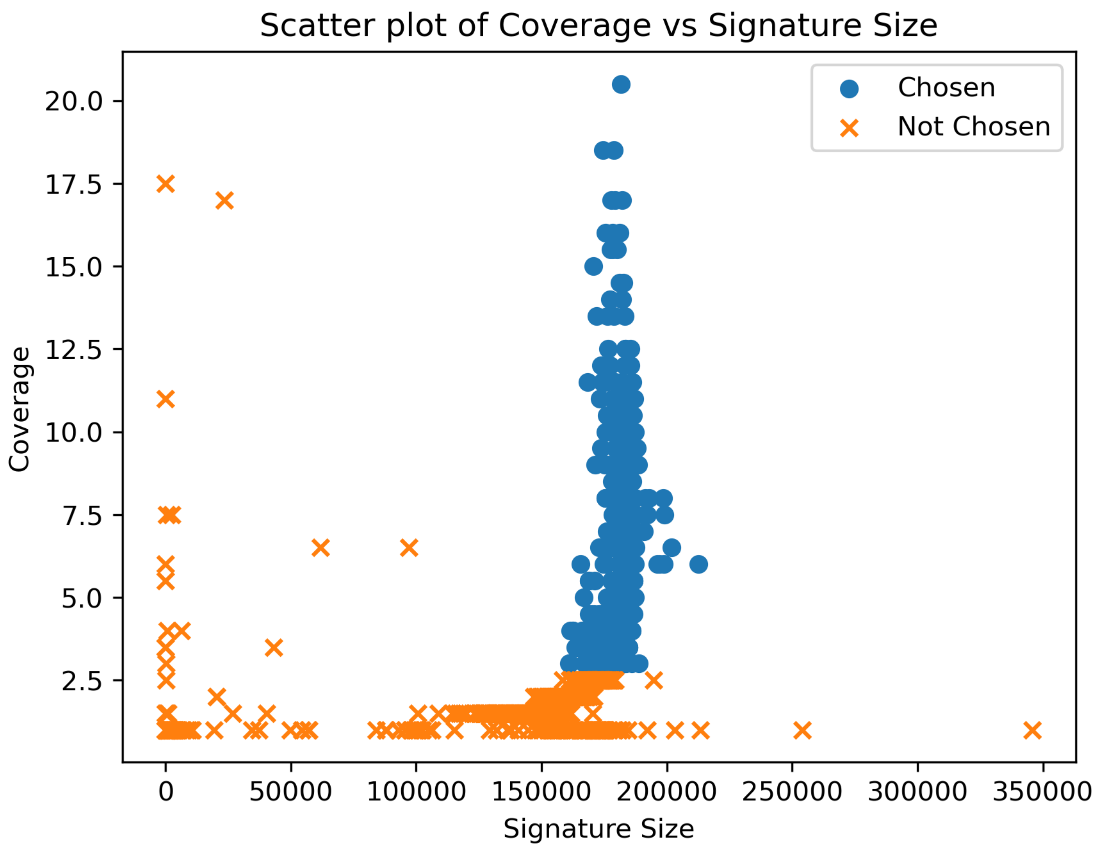

Figure 4.4. **Comparison of Sourmash Signature Size and Average Genome Coverage** The scatter plot displays the coverage and signature sizes for 1,752 samples downloaded from the SRA. We selected samples with a coverage greater than 2 and a signature size ranging between 150K and 210K.

The filtered 1,103 samples are classified under three species: 977 samples are *Bos taurus*, 52 samples are *Bos indicus*, and 75 samples belong to other species(bosoutgroup).

Out of the *Bos taurus* samples, 768 represent 80 different breeds, while 209 samples lack a breed annotation in their metadata and are named 'taurus'. Figure 4.5 displays the 15 breeds that are each represented by 9 or more samples.

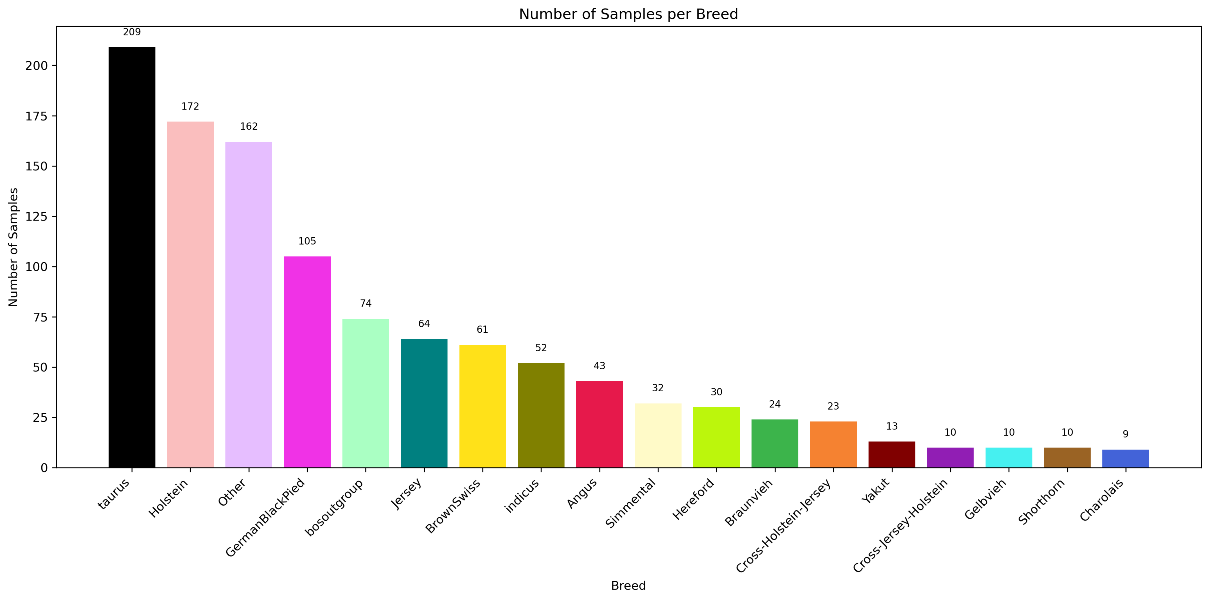

Figure 4.5. **Histogram of Breeds Represented in the CCDG** The histogram presents the count of samples for each breed within the CCDG. We have restricted the histogram to display only the 14 breeds that comprise 9 or more animals. All remaining samples are grouped under the 'Other' category, which collectively represents 66 breeds.

We used kSpider to cluster the filtered samples using their sourmash signatures, and the resultant dendrogram visualized by iTOL [82] is displayed in Figure 4.6. Both the outgroup and *Bos indicus* form distinct clusters and are separated from the rest of the *Bos taurus* samples. However, a few *Bos taurus* samples are clustered with the *Bos indicus* samples. Moreover, we can use the clustering data to determine the missing breed in the metadata. Most samples with missing breeds fall in the same cluster as the Brown Swiss and Holstein breeds.

Samples with missing breeds are distributed over all clusters but mostly clustered with Brown Swiss and Holstein Breeds. Determining the breed of those samples is possible from the dendrogram.

We established seven clusters from the filtered samples: five for *Bos taurus* samples and two for Indicus and Bosoutgroup. Detailed information about these clusters can

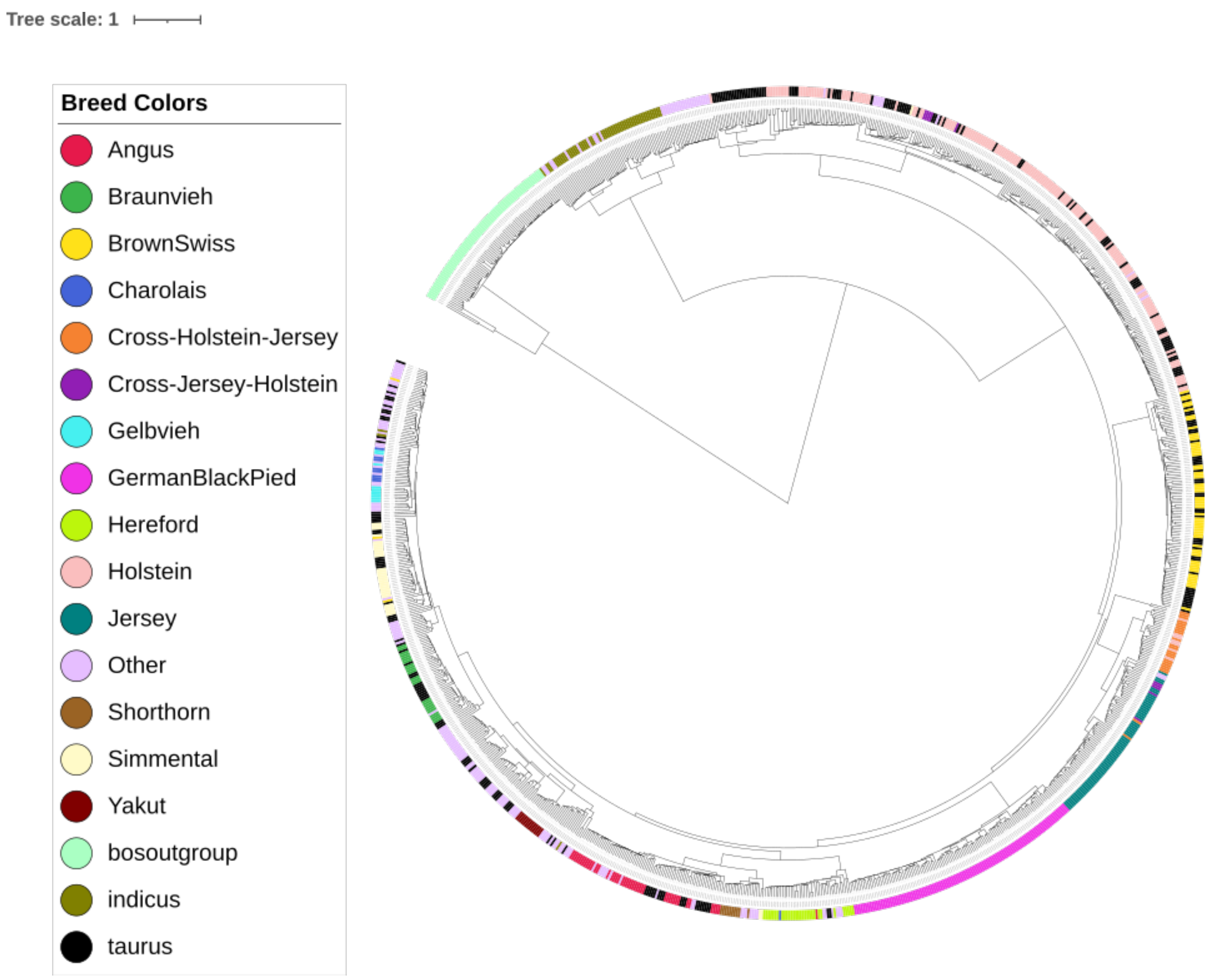

Figure 4.6. **Samples dendrogram** The dendrogram displays 1,103 samples. We colored the samples of 14 breeds. We labeled 162 samples as "Other", which represent 66 different breeds. Additionally, 209 samples lack a breed name in their metadata and were placed under the 'taurus' category.

be found in Table 4.2. An independent CCDG was constructed for each cluster using metagraph. The combined size of all CCDGs is 293.8 GB, encompassing 1,103 samples. On average, each cluster contains approximately 7 billion kmers.

### 4.3.3 Creating a Catalog of Common Structural Variants in *Bos taurus* Breeds

We called SVs from 26 breeds using a workflow described in Figure 4.2 and results are summarized in table 4.3. We also merged the SV catalog from the *Bos indicus* pangenome [20] to broaden the scope of our common catalog. We identified 1.5 million SVs, of which 1

| Cluster Name | Size(Gb) | Number of samples | Number of kmers |
|---|---|---|---|
| cluster.taurus.1 | 57.8 | 199 | 8,482,368,671 |
| cluster.taurus.2 | 46 | 198 | 7,129,180,530 |
| cluster.taurus.3 | 54.3 | 200 | 7,568,824,104 |
| cluster.taurus.4 | 47 | 200 | 7,076,545,430 |
| cluster.taurus.5 | 55 | 180 | 8,900,797,784 |
| indicus | 14.1 | 52 | 5,614,933,835 |
| bosoutgroup | 19.6 | 74 | 6,396,304,073 |

Table 4.2. **CCDG clusters data** The samples are organized into seven clusters, with a CCDG created independently for each cluster. The table details the characteristics of each CCDG.

million had an allele frequency greater than 0.05. Using Jasmine, we merged the 1 million variants and discovered 175,540 unique SVs. Trio assemblies produced the most unique variants, which confirms our expectations from Figure 4.3. Additionally, 67% of the variants in the indicus SV catalog had an allele frequency of less than 0.05. In conclusion, the catalog of common SVs contains 175K variants, which are filtered and compiled from an original call set of 1.5M SVs.

| **Source** | **Number of Breeds** | **Number of Variants** | **Number of Variants AF > 0.05** | **Number of Unique Variants AF > 0.05** |
|---|---|---|---|---|
| Trio Assemblies | 7 | 234,577 | 95,618 | 55,291 |
| LRS Callers | 8 | 957,079 | 337,609 | 20,507 |
| Assemblies | 10 | 164,487 | 97,394 | 14,993 |
| Indicus SV Catalog | 10 | 107,639 | 35,895 | 6,536 |
| SRS Callers | 8 | 74,443 | 53,742 | 2,720 |
| Total | 26 | 1,538,225 | 620,303 | 175,540 |

Table 4.3. **Quantitative Breakdown of Structural Variants at Each Stage of Catalog Compilation** We called SVs from 5 sources, and the table reports the number of variants each source called and how many of them are common (AF > 0.05). The last column represents the unique contribution of each source to the final catalog.

The catalog encompasses a diverse array of SVs, including 54K deletions, 59K insertions, and 61K complex variants, as evidenced by the size distribution histogram in Figure 4.7. Most SVs are under 10K in size, while a subset of 30 SVs are notably larger than 100K, having been detected through assembly-based calling and long-read sequencing techniques. This demonstrates the catalog's comprehensive coverage of variations across different sizes and types, providing a robust resource for genomic studies in *Bos taurus* breeds.

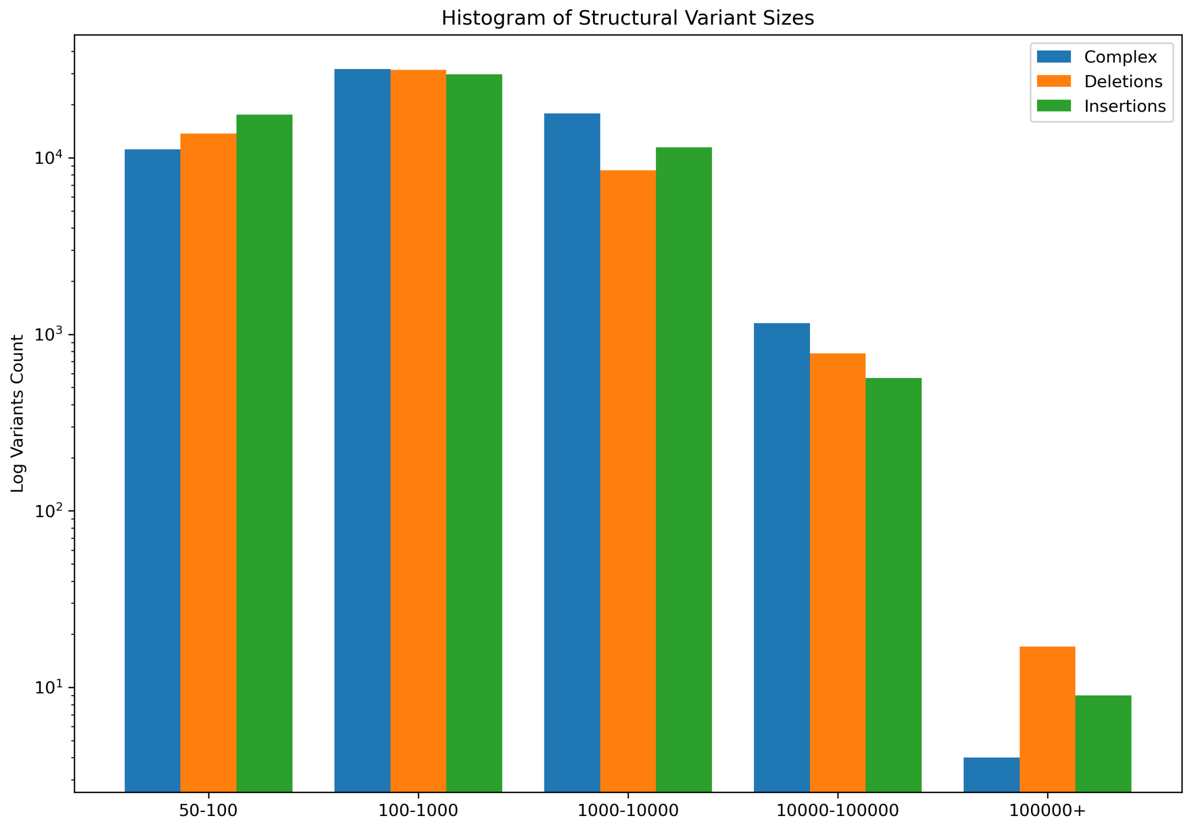

Figure 4.7. **Structural Variants Sizes Histogram** The histogram in the figure displays the distribution of SV sizes, categorized by type. The y-axis is on a logarithmic scale to facilitate the visualization of low-occurrence SVs larger than 100K

We improved the population genotyping recall by imputing genotypes of common SVs using Beagle. We created a reference panel using the 175K SVs and 84M SNVs from the Ensembl catalog (v110). All variants are genotyped in all the 977 Taurus samples. Beagle imputed missing genotypes in the reference panel and phased the variants to form a comprehensive reference panel for all variations. We calculated the allele frequency using the improved reference panel. Figure 4.8 shows the distribution of allele frequencies for SVs stratified by the main breeds. All main breeds are represented evenly by the SVs in the catalog.

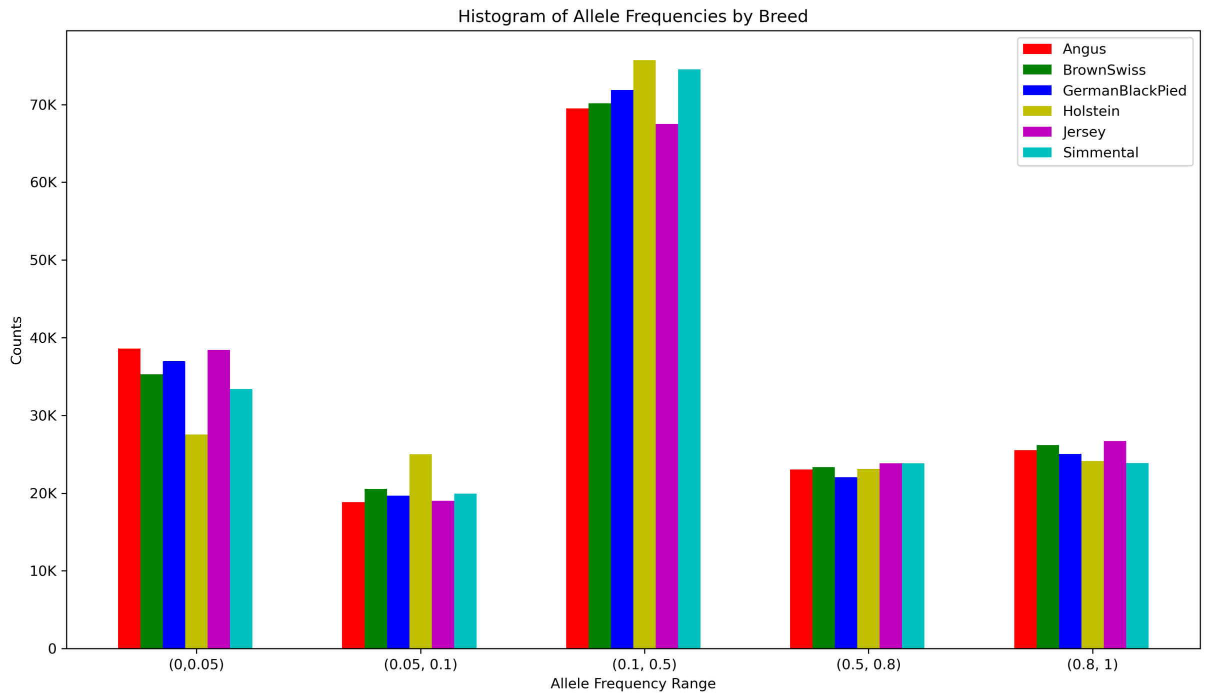

Figure 4.8. **Allele Frequency Per Breed Histogram** Figure shows the histogram of allele frequencies stratified by main breeds.

We employed the Variant Effect Predictor (VEP) [110] to assess the functional effects of the discovered common SV variants. Our analysis revealed that these SVs intersect with 15K genes and 26K transcripts. The potential consequences of the SVs as predicted by VEP are compiled in Table 4.4. Approximately 4.6K variants are deemed to likely exert a high impact on gene function due to various consequences, such as frameshifts or transcript ablation. Another 2.8K SVs are anticipated to have moderate or low impacts on gene function. The functional implications for the remaining variants cannot be determined by VEP. These are located within intronic sequences or represent intergenic variants. Ultimately, functional annotations have been integrated into the catalog to enhance its utility.

| Consequence | Number of Variants | Description | Impact |
|---|---|---|---|
| transcript_ablation | 968 | A feature ablation whereby the deleted region includes a transcript feature | HIGH |
| splice_acceptor_variant | 1,572 | A splice variant that changes the 2 base region at the 3' end of an intron | HIGH |
| splice_donor_variant | 1,829 | A splice variant that changes the 2 base region at the 5' end of an intron | HIGH |
| stop_gained | 223 | A sequence variant whereby at least one base of a codon is changed, resulting in a premature stop codon, leading to a shortened transcript | HIGH |
| frameshift_variant | 372 | A sequence variant which causes a disruption of the translational reading frame, because the number of nucleotides inserted or deleted is not a multiple of three | HIGH |
| inframe_insertion | 210 | An inframe non synonymous variant that inserts bases into in the coding sequence | MODERATE |
| inframe_deletion | 313 | An inframe non synonymous variant that deletes bases from the coding sequence | MODERATE |
| missense_variant | 40 | A sequence variant, that changes one or more bases, resulting in a different amino acid sequence but where the length is preserved | MODERATE |
| splice_donor_5th_base_variant | 1,796 | A sequence variant that causes a change at the 5th base pair after the start of the intron in the orientation of the transcript | LOW |
| splice_region_variant | 121 | A sequence variant in which a change has occurred within the region of the splice site, either within 1-3 bases of the exon or 3-8 bases of the intron | LOW |
| splice_donor_region_variant | 87 | A sequence variant that falls in the region between the 3rd and 6th base after splice junction (5' end of intron) | LOW |
| splice_polypyrimidine_tract_variant | 271 | A sequence variant that falls in the polypyrimidine tract at 3' end of intron between 17 and 3 bases from the end (acceptor -3 to acceptor -17) | LOW |
| coding_sequence_variant | 2,350 | A sequence variant that changes the coding sequence | MODIFIER |
| 5_prime_UTR_variant | 1,345 | A UTR variant of the 5' UTR | MODIFIER |
| 3_prime_UTR_variant | 1,222 | A UTR variant of the 3' UTR | MODIFIER |
| non_coding_transcript_exon_variant | 323 | A sequence variant that changes non-coding exon sequence in a non-coding transcript | MODIFIER |
| intron_variant | 124,253 | A transcript variant occurring within an intron | MODIFIER |
| non_coding_transcript_variant | 2,836 | A transcript variant of a non coding RNA gene | MODIFIER |
| coding_transcript_variant | 230 | A transcript variant of a protein coding gene | MODIFIER |
| upstream_gene_variant | 13,176 | A sequence variant located 5' of a gene | MODIFIER |
| downstream_gene_variant | 12,930 | A sequence variant located 3' of a gene | MODIFIER |
| intergenic_variant | 106,656 | A sequence variant located in the intergenic region, between genes | MODIFIER |

Table 4.4. **Summary of variant consequences and their impact.** The table summarizes the output from the Variant Effect Predictor (VEP) detailing the predicted consequences of structural variants. Descriptions of each consequence are sourced from the VEP online guide [111]

## 4.4 Discussion

Uncovering high-quality, common SVs in *Bos taurus* breeds is a challenge that has not yet been fully solved. Our method tackles this by taking advantage of all SV sources. We genotyped variants in a large set of 977 samples. This helped us to weed out errors and rare variants and to work out important statistics.

We began by evaluating the efficacy of SV calling methods in cattle, utilizing data from the Brahman and Angus trio assembly project [105]. SVs were identified from SRS, LRS, and haploid-resolved assemblies employing various SV callers. This experiment revealed that each technique offers a unique perspective. Combining the results of the different methods is important to achieve the best recall. Nevertheless, such integration of results leads to a decline in precision, necessitating the implementation of filtering strategies to purge false positives. The outcomes of our experiments guided the development of a workflow for establishing an extensive catalog of common SVs for *Bos taurus* breeds with high precision.

We compiled a list of candidate SVs in various *Bos taurus* breeds. This list initially consisted of 1.4 million SVs, called from 10 refseq assemblies, 4 trio assemblies, SRS, and LRS samples. These samples represented 16 breeds. Additionally, we incorporated 100K SVs from the Indicus catalog into the list to form an initial call set of 1.5 millions SVs.

Genotyping the variants in a larger number of *Bos taurus* samples is a critical step in our workflow. For this task, we employed The Great Genotyper to perform population-scale genotyping. The workflow of The Great Genotyper begins by creating a Counting Colored De Bruijn Graph (CCDG) for the population, which is then used to genotype a list of variants within this population.

To create the CCDG, we initially collected 1700 samples from the Sequence Read Archive (SRA). We then applied sourmash to screen for and exclude contaminated samples, resulting in 1100 samples, of which 977 were *Bos taurus* samples. From these, we generated a CCDG that is 293.8 GB in size. This CCDG played a pivotal role in the creation of our catalog and offers a resource that can be reused in future studies for efficient population genotyping.

Subsequently, we genotyped the collective 1.5 million variants in 977 *Bos taurus* samples utilizing a tool we refer to as The Great Genotyper. We utilized Beagle to impute missing genotypes due to lack of coverage. To ensure that Beagle could accurately identify the haplotypes and enhance its imputation capabilities, we augmented the SVs with single nucleotide variants (SNVs) from the Ensembl catalog, after genotyping these in the 977 samples. This process enabled us to calculate the allele frequency for the newly discovered SVs. We then applied a filtering criterion, eliminating SVs with an allele frequency below 0.05, as they are likely to be either rare variants or false positives.

The refinement process continued with the merging of the filtered list using the Jasmine [112] resulting in a pool of 175K unique variants, each characterized by their respective allele frequency within the 977 samples. VEP [110] predicted that those SVs affect 15K genes and 26K transcripts, with an estimated 4.6K variants presumed to exert a high impact on gene function.

Our SV catalog outperforms previous attempts by containing a higher number of high-quality common SVs, and the variants have been genotyped in significantly more samples. This improvement is primarily due to The Great Genotyper's efficiency in population genotyping. Our ability to calculate allele frequency swiftly and accurately enabled us to filter out rare variants and errors. This allowed us to be very inclusive when initially calling SVs for the catalog and to merge all SV sources. Consequently, we identified 175K SVs that can be genotyped in at least 48 animals. Notably, 2.7K of these variants were detected using SRS alone, which is generally associated with lower calling quality. Additionally, by genotyping these variants along with SNVs from the Ensembl catalog, we have set the stage for imputation, which can further improve the population genotyping outcomes. In summary, the integration of population genotyping with SV calling has led to the creation of a more extensive and higher-quality common SV catalog.

The current SV catalog does not cover copy number variants (CNVs), although studies have shown their association with phenotypes traits such as coat color [113]. We did not include CNVs because the current model in The Great Genotyper is unable to genotype CNVs. This limitation can be addressed by developing a dedicated CNV genotyping tool that makes use of the already constructed CCDG.

Our catalog is designed for straightforward expansion, as The Great Genotyper can calculate allele frequencies for any new set of SVs in the 977 samples represented in the CCDG in hours. With the decreasing cost of LRS, more laboratories are likely to sequence additional breeds, leading to more assemblies or mapping-based variant studies. Common

SVs can be identified from those samples to enhance breed coverage. Additionally, incorporating SVs from new SRS samples will be valuable, as they will broaden the coverage across more breeds while mitigating the concern of introducing more errors. In summary, our method for creating the SV catalog is scalable and well-suited to keep up with the pace of cattle sequencing projects.

The catalog provides a crucial resource for understanding the genomic diversity among different breeds, which may shed light on the genetic basis of complex traits. This understanding can inform breeding strategies to enhance cattle adaptation to specific environments and increase their economic value. Additionally, the catalog can assist in developing breed-specific pangenomes. It can offer breed-stratified allele frequencies for pangenome variants, which can be used to streamline the pangenome structure and enhance variant analysis based on these pangenomes.

Our workflow for discovering high-quality, common SVs in cattle breeds stands on two pillars: First, we call SVs from every available source to maximize recall, accepting the risk of false positives. Second, we leverage the efficiency of The Great Genotyper to calculate allele frequencies by genotyping the variants in a population of 977 samples. We then use the allele frequency data to filter out the false positives and rare variants. This workflow has produced a catalog of 175,000 high-quality, common SVs spanning 80 breeds. The SVs in the catalog are enhanced with precise allele frequencies and functional annotations to improve their utility.

## 4.5 List of abbreviations

- SV: Structural Variant
- SNP: Single Nucleotide Polymorphism
- SNV: Single Nucleotide Variant
- CNV: Copy Number Variant
- SRS: Short Read Sequencing
- LRS: Long Read Sequencing

- CCDG: Counting Colored Debruijn Graph
- PAV: Phased Assembly Variant Caller
- AF: Allele Frequency
- VEP: Variant Effect Predictor
- SRA: Sequence Read Archive

# Chapter 5

# Conclusion and Future Work

Population-scale SV studies mine hundreds of terabytes of genomic sequences to characterize genomic diversity among populations. The sheer amount of data poses challenges in storing, analyzing, and aggregating. However, it also opens the door for more accurate joint analysis and a deeper understanding of genomic variation.

Here, I proposed a kmer based solution to the big data challenges of population scale SV studies. I condensed hundreds of terabytes of human and cattle populations into counting colored DeBruijn graphs(CCDG) of a size 100-fold smaller than CRAM files. Also, I provided a population genotyping workflow, The Great Genotyper, that performs joint genotyping of SV and small variants in the samples of a CCDG. The Great Genotyper solves the N+l problem, which is crucial in population SV studies. It can analyze newly discovered SVs in the population without the need to handle the hundreds of terabytes of CRAM files. The population genotyping requires only the CCDG and has the same computational requirements as variant calling, making it feasible for all genomics labs. We demonstrated that removing these limitations unleashes the potential of population's SV studies. Using the human CCDG, we showed how easy to calculate accurate allele frequencies of SVs to aid gene discovery studies. Moreover, we presented a new imputation panel using the SVs from the Human Pangenome and found strong associations between GWAS SNPs and these SVs.

Furthermore, I designed a hybrid approach of mapping-based SV discovery tools and kmer based population genotyping to create a catalog for common SV for *Bos taurus* breeds. The workflow calls variants from multiple sources, genotypes the discovered variants in the *Bos taurus* population using The Great Genotyper, and then filters out rare variants and errors using the allele frequency calculated from the population genotyping results. The created catalog covers more breeds than its competitors and contains more high-quality SVs. Additionally, the SVs are genotyped in 977 samples along with small variants, paving the way for more population-scale applications. Lastly, I developed a data structure specialized in the association of kmers to its counts and other metadata to enhance the development of DeBruijn graphs and their derivatives.

In recent decades, sequencing technologies have experienced significant advancements in two distinct areas. Firstly, Illumina has reduced the cost of short-read sequencing, enabling the scientific community to sequence a vast number of genomes. Secondly, PacBio and Oxford Nanopore have developed long-read sequencers that produce high-quality, long reads capable of generating high-quality, phased SVs and haplotype-resolved assemblies. Previous population studies utilized both technologies, but the computational methods employed were not scalable enough to keep up with the pace of sequencing. In this thesis, we address the scalability challenge to streamline the integration of knowledge from an immense number of SRS samples into newly discovered variations. We also demonstrate the newfound potential of population-scale SV studies after overcoming these challenges.

Although the population CCDG encapsulates the vast genomic diversity of thousands of samples, it has its limitations. Representing sequences solely by their kmer content results in a loss of contextual information for each kmer. Kmers originating from repeated regions tend to be collapsed into a single node within the graph, making it challenging to trace their genomic origins. This complexity hampers the processing of variants in the genome's repeated and low-complexity regions.

The issue is somewhat mitigated through the use of an imputation model based on the pangenome, where the genotypes of variants in these problematic regions can be imputed using the genotypes of nearby variants on the same haplotype. Despite this, mapping-based methods retain an edge in these regions. For instance, GATK's accuracy in genotyping small variants exceeds that of both Pangenie and The Great Genotyper, highlighting the conditions where mapping-based methods are superior to kmer based methods.

In this thesis, we demonstrate the significance of using CCDGs in population studies. Although The Great Genotyper's performance is unparalleled, it can be further improved. Firstly, a specialized CCDG can be created specifically for The Great Genotyper, with its size reduced by omitting unused information. The Great Genotyper predicts the copy number of a kmer by comparing kmer counts to the average genome coverage. This specialized CCDG needs only to contain the results of the copy number predictions, not all kmer counts, as The Great Genotyper does not use kmers with a copy number greater than two. Therefore, the counts in the specialized CCDGs will be limited to either 1 or 2. These new CCDGs can be compressed more effectively since the kmer counts will predominantly consist of 1s or 2s. Additionally, new data structures can be developed to meet this requirement. For instance, we can create two distinct CCDGs for the same samples, each containing kmers of the same copy number. These new CCDGs can further reduce size and decrease the memory requirements for The Great Genotyper, enabling the creation of CCDGs for hundreds of thousands or even millions of samples.

Moreover, the genotyping model for The Great Genotyper can be expanded to encompass more types of structural variants. While copy number variants can be genotyped using kmer counts, this process is complex as it now requires consideration of kmers with copy numbers greater than two, which may originate from different genome regions unrelated to the variants being genotyped. However, we can leverage population information using approaches similar to those in muCNV [67]. Samples can be clustered based on the counts of kmers representing the copy number variations, and the copy number can be jointly determined for each sample cluster. I hypothesize that utilizing information from multiple samples will aid in accurately determining the correct copy number.

CCDGs hold a wealth of information about genome diversity, and their functionality extends beyond population genotyping. Numerous applications can be developed to mine these indexed genomes. For instance, an SV calling algorithm can leverage the CCDG to identify regions in the graph that diverge from the reference, and then assemble all alternate sequences present in the samples. Additionally, CCDGs can aid in creating pangenomes. Pangenome graphs can become highly complex due to variations among input genomes, leading to intricate graphs with many nodes. Removing nodes corresponding to rare variants can simplify the pangenome without diminishing its effectiveness. The CCDG can be consulted during pangenome construction to simplify the graph while maintaining its representativeness. Furthermore, population-specific pangenomes can be developed by filtering out parts of the pangenome that are not present in a particular population's CCDG. These population-specific pangenomes can be simpler, potentially improving the mappability of reads to the pangenome.

Tools can be developed to utilize the created SV catalog for cattle breeds. Genotyping and imputing these variants is essential, particularly with the increasing use of low-pass sequencing. I propose creating a genotyping and imputation algorithm similar to Pangenie. Pangenie genotypes variants from the pangenome by extracting kmers specific to the variants for genotyping, and it utilizes the pangenome's phasing information to impute variants without kmer support. My proposal involves developing a similar algorithm for the SV catalog, where the genotyping process will be identical to that of Pangenie. However, the phasing information will be substituted with co-occurrence statistics between kmers , calculated from the CCDGs. This new algorithm will assist in identifying common variants in the catalog in new samples, even those sequenced with low coverage.

In conclusion, thousands of SRS samples can be succinctly represented in CCDGs, enabling efficient kmer searching. CCDGs facilitate population genotyping for SV, elevating genomic studies to new heights. We have developed algorithms and data structures for utilizing CCDGs in population studies, with prime examples in human and cattle genomics. Further exploration of CCDGs could address additional scalability challenges in this field.

# References

[1] K. Katz, O. Shutov, R. Lapoint, M. Kimelman, J. R. Brister, and C. O'Sullivan, "The Sequence Read Archive: A decade more of explosive growth," vol. 50, no. D1, pp. D387–D390. [Online]. Available: https://doi.org/10.1093/nar/gkab1053

[2] F. K. Mastrorosa, D. E. Miller, and E. E. Eichler, "Applications of long-read sequencing to Mendelian genetics," vol. 15, no. 1, p. 42. [Online]. Available: https://doi.org/10.1186/s13073-023-01194-3

[3] L. H. Goetz and N. J. Schork, "Personalized Medicine: Motivation, Challenges and Progress," vol. 109, no. 6, pp. 952–963. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6366451/

[4] J. Crossa, R. Fritsche-Neto, O. A. Montesinos-Lopez, G. Costa-Neto, S. Dreisigacker, A. Montesinos-Lopez, and A. R. Bentley, "The Modern Plant Breeding Triangle: Optimizing the Use of Genomics, Phenomics, and Enviromics Data," vol. 12. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fpls.2021.651480

[5] M. Johnsson, "Genomics in animal breeding from the perspectives of matrices and molecules," vol. 160, no. 1, p. 20.

[6] M. Mahmoud, N. Gobet, D. I. Cruz-Dávalos, N. Mounier, C. Dessimoz, and F. J. Sedlazeck, "Structural variant calling: The long and the short of it," vol. 20, no. 1, p. 246. [Online]. Available: https://genomebiology.biomedcentral.com/articles/10.1186/s13059-019-1828-7

[7] H. Li, "Minimap2: Pairwise alignment for nucleotide sequences," vol. 34, no. 18, pp. 3094–3100. [Online]. Available: https://github.com/ruanjue/smartdenovo;

[8] H. Li and R. Durbin, "Making the Leap: Maq to BWA," vol. 25, no. 14, pp. 1754–1760. [Online]. Available: http://massgenomics.org/2009/12/making-the-leap-maq-to-bwa.html

[9] X. Chen, O. Schulz-Trieglaff, R. Shaw, B. Barnes, F. Schlesinger, M. Källberg, A. J. Cox, S. Kruglyak, and C. T. Saunders, "Manta: Rapid detection of structural variants and indels for germline and cancer sequencing applications," vol. 32, no. 8, pp. 1220–1222. [Online]. Available: https://academic.oup.com/bioinformatics/article/32/8/1220/1743909

[10] T. Rausch, T. Zichner, A. Schlattl, A. M. Stütz, V. Benes, and J. O. Korbel, "DELLY: Structural variant discovery by integrated paired-end and split-read analysis," vol. 28, no. 18, pp. 333–339. [Online]. Available: https://academic.oup.com/bioinformatics/article/28/18/i333/245403

[11] T. Jiang, Y. Liu, Y. Jiang, J. Li, Y. Gao, Z. Cui, Y. Liu, B. Liu, and Y. Wang, "Long-read-based human genomic structural variation detection with cuteSV," vol. 21, no. 1, p. 189. [Online]. Available: https://doi.org/10.1186/s13059-020-02107-y

[12] S. Zarate, A. Carroll, M. Mahmoud, O. Krasheninina, G. Jun, W. J. Salerno, M. C. Schatz, E. Boerwinkle, R. A. Gibbs, and F. J. Sedlazeck, "Parliament2: Accurate structural variant calling at scale," vol. 9, no. 12, p. giaa145. [Online]. Available: https://doi.org/10.1093/gigascience/giaa145

[13] F. J. Sedlazeck, P. Rescheneder, M. Smolka, H. Fang, M. Nattestad, A. Von Haeseler, and M. C. Schatz, "Accurate detection of complex structural variations using single-molecule sequencing," vol. 15, no. 6, pp. 461–468. [Online]. Available: /pmc/articles/PMC5990442/?report=abstract

[14] G. Jun, A. C. English, G. A. Metcalf, J. Yang, M. J. Chaisson, N. Pankratz, V. K. Menon, W. J. Salerno, O. Krasheninina, A. V. Smith, J. A. Lane, T. Blackwell, H. M. Kang, S. Salvi, Q. Meng, H. Shen, D. Pasham, S. Bhamidipati, K. Kottapalli, D. K. Arnett, A. Ashley-Koch, P. L. Auer, K. M. Beutel, J. C. Bis, J. Blangero, D. W. Bowden, J. A. Brody, B. E. Cade, Y.-D. I. Chen, M. H. Cho, J. E. Curran, M. Fornage, B. I. Freedman, T. Fingerlin, B. D. Gelb, L. Hou, Y.-J. Hung, J. P. Kane, R. Kaplan, W. Kim, R. J. F. Loos, G. M. Marcus, R. A. Mathias, S. T. McGarvey, C. Montgomery, T. Naseri, S. M. Nouraie, M. H. Preuss, N. D. Palmer, P. A. Peyser, L. M. Raffield, A. Ratan, S. Redline, S. Reupena, J. I. Rotter, S. S. Rich, M. Rienstra, I. Ruczinski, V. G. Sankaran, D. A. Schwartz, C. E. Seidman, J. G. Seidman, E. K. Silverman, J. A. Smith, A. Stilp, K. D. Taylor, M. J. Telen, S. T. Weiss, L. K. Williams, B. Wu, L. R. Yanek, Y. Zhang, J. Lasky-Su, M. C. Gingras, S. K. Dutcher, E. E. Eichler, S. Gabriel, S. Germer, R. Kim, K. A. Viaud-Martinez, D. A. Nickerson, N. T.-O. f. P. M. T. Consortium, J. Luo, A. Reiner, R. A. Gibbs, E. Boerwinkle, G. Abecasis, and F. J. Sedlazeck. Structural variation across 138,134 samples in the TOPMed consortium. [Online]. Available: https://www.biorxiv.org/content/10.1101/2023.01.25.525428v1

[15] J. R. S. Meadows, J. M. Kidd, G.-D. Wang, H. G. Parker, P. Z. Schall, M. Bianchi, M. J. Christmas, K. Bougiouri, R. M. Buckley, C. Hitte, A. K. Nguyen, C. Wang, V. Jagannathan, J. E. Niskanen, L. A. F. Frantz, M. Arumilli, S. Hundi, K. Lindblad-Toh, C. Ginja, K. K. Agustina, C. André, A. R. Boyko, B. W. Davis, M. Drögemüller, X.-Y. Feng, K. Gkagkavouzis, G. Iliopoulos, A. C. Harris, M. K. Hytönen, D. C. Kalthoff, Y.-H. Liu, P. Lymberakis, N. Poulakakis, A. E. Pires, F. Racimo, F. Ramos-Almodovar, P. Savolainen, S. Venetsani, I. Tammen, A. Triantafyllidis, B. vonHoldt, R. K. Wayne, G. Larson, F. W. Nicholas, H. Lohi, T. Leeb, Y.-P. Zhang, and E. A. Ostrander, "Genome sequencing of 2000 canids by the Dog10K consortium advances the understanding of demography, genome function and architecture," vol. 24, no. 1, p. 187. [Online]. Available: https://doi.org/10.1186/s13059-023-03023-7

[16] C. Quan, H. Lu, Y. Lu, and G. Zhou, "Population-scale genotyping of structural variation in the era of long-read sequencing," vol. 20, pp. 2639–2647. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2001037022002033

[17] D. Y. C. Brandt, V. R. C. Aguiar, B. D. Bitarello, K. Nunes, J. Goudet, and D. Meyer, “Mapping Bias Overestimates Reference Allele Frequencies at the HLA Genes in the 1000 Genomes Project Phase I Data,” vol. 5, no. 5, pp. 931–941. [Online]. Available: https://doi.org/10.1534/g3.114.015784

[18] J. F. Degner, J. C. Marioni, A. A. Pai, J. K. Pickrell, E. Nkadori, Y. Gilad, and J. K. Pritchard, “Effect of read-mapping biases on detecting allele-specific expression from RNA-sequencing data,” vol. 25, no. 24, pp. 3207–3212. [Online]. Available: https://doi.org/10.1093/bioinformatics/btp579

[19] T. Günther and C. Nettelblad, “The presence and impact of reference bias on population genomic studies of prehistoric human populations,” vol. 15, no. 7, p. e1008302. [Online]. Available: https://journals.plos.org/plosgenetics/article?id=10.1371/journal.pgen.1008302

[20] X. Dai, P. Bian, D. Hu, F. Luo, Y. Huang, S. Jiao, X. Wang, M. Gong, R. Li, Y. Cai, J. Wen, Q. Yang, W. Deng, H. A. Nanaei, Y. Wang, F. Wang, Z. Zhang, B. D. Rosen, R. Heller, and Y. Jiang, “A Chinese indicine pangenome reveals a wealth of novel structural variants introgressed from other Bos species,” vol. 33, no. 8, pp. 1284–1298. [Online]. Available: https://genome.cshlp.org/content/33/8/1284

[21] B. D. Rosen, D. M. Bickhart, R. D. Schnabel, S. Koren, C. G. Elsik, E. Tseng, T. N. Rowan, W. Y. Low, A. Zimin, C. Couldrey, R. Hall, W. Li, A. Rhie, J. Ghurye, S. D. McKay, F. Thibaud-Nissen, J. Hoffman, B. M. Murdoch, W. M. Snelling, T. G. McDaneld, J. A. Hammond, J. C. Schwartz, W. Nandolo, D. E. Hagen, C. Dreischer, S. J. Schultheiss, S. G. Schroeder, A. M. Phillippy, J. B. Cole, C. P. Van Tassell, G. Liu, T. P. L. Smith, and J. F. Medrano, “De novo assembly of the cattle reference genome with single-molecule sequencing,” vol. 9, no. 3, p. giaa021. [Online]. Available: https://doi.org/10.1093/gigascience/giaa021

[22] D. S. Standage, C. T. Brown, and F. Hormozdiari, “Kevlar: A Mapping-Free Framework for Accurate Discovery of De Novo Variants,” vol. 18, pp. 28–36. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6682328/

[23] J. Ebler, P. Ebert, W. E. Clarke, T. Rausch, P. A. Audano, T. Houwaart, Y. Mao, J. O. Korbel, E. E. Eichler, M. C. Zody, A. T. Dilthey, and T. Marschall, “Pangenome-based genome inference allows efficient and accurate genotyping across a wide spectrum of variant classes,” vol. 54, no. 4, pp. 518–525. [Online]. Available: https://www.nature.com/articles/s41588-022-01043-w

[24] P. Khorsand and F. Hormozdiari, “Nebula: Ultra-efficient mapping-free structural variant genotyper,” vol. 49, no. 8, p. e47. [Online]. Available: https://doi.org/10.1093/nar/gkab025

[25] J. Fan, N. P. Singh, J. Khan, G. E. Pibiri, and R. Patro. Fulgor: A fast and compact k-mer index for large-scale matching and color queries. [Online]. Available: https://www.biorxiv.org/content/10.1101/2023.05.09.539895v1

[26] T. Kolajo, O. Daramola, and A. Adebiyi, "Big data stream analysis: A systematic literature review," vol. 6, no. 1, p. 47. [Online]. Available: https://doi.org/10.1186/s40537-019-0210-7

[27] "Matias Y, Vitter JS, Young NE. Approximate data structures with applications. In: Proceedings of the fifth annual ACM-SIAM symposium on discrete algorithms. Arlington: Society for Industrial and Applied Mathematics; 1994. p. 187–194."

[28] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," vol. 13. [Online]. Available: https://doi.org/10.1145/362686.362692

[29] P. Pandey, M. A. Bender, R. Johnson, and R. Patro, "A General-Purpose Counting Filter," pp. 775–787.

[30] S. C. Manekar and S. R. Sathe, "A benchmark study of k-mer counting methods for high-throughput sequencing," vol. 7, no. 12.

[31] Y. Yu, "SeqOthello: Querying RNA-seq experiments at scale," vol. 19. [Online]. Available: https://doi.org/10.1186/s13059-018-1535-9

[32] G. Cormode and S. Muthukrishnan, "An improved data stream summary: The count-min sketch and its applications," vol. 55, no. 1, pp. 58–75.

[33] N. L. Bray, H. Pimentel, P. Melsted, and L. Pachter, "Near-optimal probabilistic RNA-seq quantification," vol. 34, no. 5, pp. 525–527.

[34] Z. Iqbal, M. Caccamo, I. Turner, P. Flicek, and G. McVean, "De novo assembly and genotyping of variants using colored de Bruijn graphs," vol. 44. [Online]. Available: https://doi.org/10.1038/ng.1028

[35] M. D. Muggli, A. Bowe, N. R. Noyes, P. S. Morley, K. E. Belk, R. Raymond, T. Gagie, S. J. Puglisi, and C. Boucher, "Succinct colored de Bruijn graphs," vol. 33, no. 20, pp. 3181–3187. [Online]. Available: https://academic.oup.com/bioinformatics/article/33/20/3181/2995815

[36] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. S. Stein, *Introduction to Algorithms*. MIT Press.

[37] G. Marçais and C. Kingsford, "A fast, lock-free approach for efficient parallel counting of occurrences of k-mers," vol. 27, no. 6, pp. 764–770.

[38] "Wang J, Chen S, Dong L, Wang G. CHTKC: A robust and efficient k-mer counting algorithm based on a lock-free chaining hash table. Brief Bioinform. 2020. https://doi.org/10.1093/bib/bbaa063."

[39] D. Belazzougui, F. C. Botelho, and M. Dietzfelbinger, "Hash, Displace, and Compress," in *Algorithms - ESA 2009*, ser. Lecture Notes in Computer Science, A. Fiat and P. Sanders, Eds. Springer, pp. 682–693.

[40] A. Limasset, G. Rizk, R. Chikhi, and P. Peterlongo. Fast and scalable minimal perfect hashing for massive key sets. [Online]. Available: http://arxiv.org/abs/1702.03154

[41] M. R. Crusoe, H. F. Alameldin, S. Awad, E. Boucher, A. Caldwell, R. Cartwright, A. Charbonneau, B. Constantinides, G. Edvenson, S. Fay, J. Fenton, T. Fenzl, J. Fish, L. Garcia-Gutierrez, P. Garland, J. Gluck, I. González, S. Guermond, J. Guo, A. Gupta, J. R. Herr, A. Howe, A. Hyer, A. Härpfer, L. Irber, R. Kidd, D. Lin, J. Lippi, T. Mansour, P. McA'Nulty, E. McDonald, J. Mizzi, K. D. Murray, J. R. Nahum, K. Nanlohy, A. J. Nederbragt, H. Ortiz-Zuazaga, J. Ory, J. Pell, C. Pepe-Ranney, Z. N. Russ, E. Schwarz, C. Scott, J. Seaman, S. Sievert, J. Simpson, C. T. Skennerton, J. Spencer, R. Srinivasan, D. Standage, J. A. Stapleton, S. R. Steinman, J. Stein, B. Taylor, W. Trimble, H. L. Wiencko, M. Wright, B. Wyss, Q. Zhang, e. zyme, and C. T. Brown, "The khmer software package: Enabling efficient nucleotide sequence analysis," vol. 4, p. 900. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4608353/

[42] J. M. Zook and M. Salit, "Genomes in a bottle: Creating standard reference materials for genomic variation—why, what and how?" vol. 12. [Online]. Available: https://doi.org/10.1186/gb-2011-12-s1-p31

[43] "Flajolet P, Fusy R, Gandouet O, Meunier F. HyperLogLog: The analysis of a near-optimal cardinality estimation algorithm. In: Discrete mathematics & theoretical computer science; 2007. p. 137–56."

[44] H. Mohamadi, H. Khan, and I. Birol, "ntCard: A streaming algorithm for cardinality estimation in genomics data," vol. 33, no. 9, pp. 1324–1330.

[45] P. Pandey, F. Almodaresi, M. A. Bender, M. Ferdman, R. Johnson, and R. Patro, "Mantis: A Fast, Small, and Exact Large-Scale Sequence-Search Index," vol. 7, no. 2, pp. 201–207.e4.

[46] T. Schlick, "Adventures with RNA graphs," vol. 143. [Online]. Available: https://doi.org/10.1016/j.ymeth.2018.03.009

[47] J. I. Sohn and J. W. Nam, "The present and future of de novo whole-genome assembly," vol. 19.

[48] S. Muthukrishnan, "Data streams: Algorithms and applications," vol. 1. [Online]. Available: https://doi.org/10.1561/0400000002

[49] R. Dementiev, L. Kettner, and P. Sanders, "STXXL: Standard template library for XXL data sets," vol. 38. [Online]. Available: https://doi.org/10.1002/spe.844

[50] "Powers D. Applications and explanations of Zipf's Law. In: CoNLL; 1998."

[51] W. Huang, L. Li, J. R. Myers, and G. T. Marth, "ART: A next-generation sequencing read simulator," vol. 28. [Online]. Available: https://doi.org/10.1093/bioinformatics/btr708

[52] H. A. Lewin, G. E. Robinson, W. J. Kress, W. J. Baker, J. Coddington, K. A. Crandall, R. Durbin, S. V. Edwards, F. Forest, M. T. P. Gilbert, M. M. Goldstein, I. V. Grigoriev, K. J. Hackett, D. Haussler, E. D. Jarvis, W. E. Johnson, A. Patrinos, S. Richards, J. C. Castilla-Rubio, P. S. Soltis, X. Xu, H. Yang, and G. Zhang, "Earth BioGenome Project: Sequencing life for the future of life," vol. 115, no. 17, pp. 4325–4333. [Online]. Available: https://www.pnas.org/doi/10.1073/pnas.1720115115

[53] K.-P. Koepfli and B. Paten, "The Genome 10K Project: A Way Forward," vol. 3, no. 1, pp. 57–111. [Online]. Available: https://doi.org/10.1146/annurev-animal-090414-014900

[54] C. Quan, Y. Li, X. Liu, Y. Wang, J. Ping, Y. Lu, and G. Zhou, "Characterization of structural variation in Tibetans reveals new evidence of high-altitude adaptation and introgression," vol. 22, no. 1, p. 159. [Online]. Available: https://doi.org/10.1186/s13059-021-02382-3

[55] K. Fujinami, R. W. Strauss, J. P.-W. Chiang, I. S. Audo, P. S. Bernstein, D. G. Birch, S. M. Bomotti, A. V. Cideciyan, A.-M. Ervin, M. J. Marino, J.-A. Sahel, S. Mohand-Said, J. S. Sunness, E. I. Traboulsi, S. West, R. Wojciechowski, E. Zrenner, M. Michaelides, H. P. N. Scholl, ProgStar Study Group, and ProgStar Study Group, "Detailed genetic characteristics of an international large cohort of patients with Stargardt disease: ProgStar study report 8," vol. 103, no. 3, pp. 390–397.

[56] H. Mostafavi, T. Berisa, F. R. Day, J. R. B. Perry, M. Przeworski, and J. K. Pickrell, "Identifying genetic variants that affect viability in large cohorts," vol. 15, no. 9, p. e2002458. [Online]. Available: https://journals.plos.org/plosbiology/article?id=10.1371/journal.pbio.2002458

[57] A. W. Pang, J. R. MacDonald, D. Pinto, J. Wei, M. A. Rafiq, D. F. Conrad, H. Park, M. E. Hurles, C. Lee, J. C. Venter, E. F. Kirkness, S. Levy, L. Feuk, and S. W. Scherer, "Towards a comprehensive structural variation map of an individual human genome," vol. 11, no. 5, p. R52. [Online]. Available: https://doi.org/10.1186/gb-2010-11-5-r52

[58] K. Cleal and D. M. Baird, "Dysgu: Efficient structural variant calling using short or long reads," vol. 50, no. 9, p. e53. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9122538/

[59] M. Mahmoud, H. Doddapaneni, W. Timp, and F. J. Sedlazeck, "PRINCESS: Comprehensive detection of haplotype resolved SNVs, SVs, and methylation," vol. 22, no. 1, p. 268. [Online]. Available: https://doi.org/10.1186/s13059-021-02486-w

[60] H. Cheng, G. T. Concepcion, X. Feng, H. Zhang, and H. Li, “Haplotype-resolved de novo assembly using phased assembly graphs with hifiasm,” vol. 18, no. 2, pp. 170–175. [Online]. Available: https://www.nature.com/articles/s41592-020-01056-5

[61] W.-W. Liao, M. Asri, J. Ebler, D. Doerr, M. Haukness, G. Hickey, S. Lu, J. K. Lucas, J. Monlong, H. J. Abel, S. Buonaiuto, X. H. Chang, H. Cheng, J. Chu, V. Colonna, J. M. Eizenga, X. Feng, C. Fischer, R. S. Fulton, S. Garg, C. Groza, A. Guarracino, W. T. Harvey, S. Heumos, K. Howe, M. Jain, T.-Y. Lu, C. Markello, F. J. Martin, M. W. Mitchell, K. M. Munson, M. N. Mwaniki, A. M. Novak, H. E. Olsen, T. Pesout, D. Porubsky, P. Prins, J. A. Sibbesen, J. Sirén, C. Tomlinson, F. Villani, M. R. Vollger, L. L. Antonacci-Fulton, G. Baid, C. A. Baker, A. Belyaeva, K. Billis, A. Carroll, P.-C. Chang, S. Cody, D. E. Cook, R. M. Cook-Deegan, O. E. Cornejo, M. Diekhans, P. Ebert, S. Fairley, O. Fedrigo, A. L. Felsenfeld, G. Formenti, A. Frankish, Y. Gao, N. A. Garrison, C. G. Giron, R. E. Green, L. Haggerty, K. Hoekzema, T. Hourlier, H. P. Ji, E. E. Kenny, B. A. Koenig, A. Kolesnikov, J. O. Korbel, J. Kordosky, S. Koren, H. Lee, A. P. Lewis, H. Magalhães, S. Marco-Sola, P. Marijon, A. McCartney, J. McDaniel, J. Mountcastle, M. Nattestad, S. Nurk, N. D. Olson, A. B. Popejoy, D. Puiu, M. Rautiainen, A. A. Regier, A. Rhie, S. Sacco, A. D. Sanders, V. A. Schneider, B. I. Schultz, K. Shafin, M. W. Smith, H. J. Sofia, A. N. Abou Tayoun, F. Thibaud-Nissen, F. F. Tricomi, J. Wagner, B. Walenz, J. M. D. Wood, A. V. Zimin, G. Bourque, M. J. P. Chaisson, P. Flicek, A. M. Phillippy, J. M. Zook, E. E. Eichler, D. Haussler, T. Wang, E. D. Jarvis, K. H. Miga, E. Garrison, T. Marschall, I. M. Hall, H. Li, and B. Paten, “A draft human pangenome reference,” vol. 617, no. 7960, pp. 312–324. [Online]. Available: https://www.nature.com/articles/s41586-023-05896-x

[62] Y. Gao, X. Yang, H. Chen, X. Tan, Z. Yang, L. Deng, B. Wang, S. Kong, S. Li, Y. Cui, C. Lei, Y. Wang, Y. Pan, S. Ma, H. Sun, X. Zhao, Y. Shi, Z. Yang, D. Wu, S. Wu, X. Zhao, B. Shi, L. Jin, Z. Hu, Y. Lu, J. Chu, K. Ye, and S. Xu, “A pangenome reference of 36 Chinese populations,” vol. 619, no. 7968, pp. 112–121. [Online]. Available: https://www.nature.com/articles/s41586-023-06173-7

[63] Y. Zhou, L. Yang, X. Han, J. Han, Y. Hu, F. Li, H. Xia, L. Peng, C. Boschiero, B. D. Rosen, D. M. Bickhart, S. Zhang, A. Guo, C. P. V. Tassell, T. P. L. Smith, L. Yang, and G. E. Liu, “Assembly of a pangenome for global cattle reveals missing sequences and novel structural variations, providing new insights into their diversity and evolutionary history,” vol. 32, no. 8, pp. 1585–1601. [Online]. Available: https://genome.cshlp.org/content/32/8/1585

[64] R. Li, M. Gong, X. Zhang, F. Wang, Z. Liu, L. Zhang, Q. Yang, Y. Xu, M. Xu, H. Zhang, Y. Zhang, X. Dai, Y. Gao, Z. Zhang, W. Fang, Y. Yang, W. Fu, C. Cao, P. Yang, Z. A. Ghanatsaman, N. J. Negari, H. A. Nanaei, X. Yue, Y. Song, X. Lan, W. Deng, X. Wang, C. Pan, R. Xiang, E. M. Ibeagha-Awemu, P. J. S. . Heslop-Harrison, B. D. Rosen, J. A. Lenstra, S. Gan, and Y. Jiang,

"A sheep pangenome reveals the spectrum of structural variations and their effects on tail phenotypes," vol. 33, no. 3, pp. 463–477. [Online]. Available: https://genome.cshlp.org/content/33/3/463

[65] Y. Huang, J. He, Y. Xu, W. Zheng, S. Wang, P. Chen, B. Zeng, S. Yang, X. Jiang, Z. Liu, L. Wang, X. Wang, S. Liu, Z. Lu, Z. Liu, H. Yu, J. Yue, J. Gao, X. Zhou, C. Long, X. Zeng, Y.-J. Guo, W.-F. Zhang, Z. Xie, C. Li, Z. Ma, W. Jiao, F. Zhang, R. M. Larkin, R. R. Krueger, M. W. Smith, R. Ming, X. Deng, and Q. Xu, "Pangenome analysis provides insight into the evolution of the orange subfamily and a key gene for citric acid accumulation in citrus fruits," pp. 1–12. [Online]. Available: https://www.nature.com/articles/s41588-023-01516-6

[66] S. Chen, P. Krusche, E. Dolzhenko, R. M. Sherman, R. Petrovski, F. Schlesinger, M. Kirsche, D. R. Bentley, M. C. Schatz, F. J. Sedlazeck, and M. A. Eberle, "Paragraph: A graph-based structural variant genotyper for short-read sequence data," vol. 20, no. 1.

[67] G. Jun, F. Sedlazeck, Q. Zhu, A. English, G. Metcalf, H. M. Kang, Human Genome Structural Variation Consortium (HGSVC), C. Lee, R. Gibbs, and E. Boerwinkle, "muCNV: Genotyping structural variants for population-level sequencing," vol. 37, no. 14, pp. 2055–2057. [Online]. Available: https://doi.org/10.1093/bioinformatics/btab199

[68] H. P. Eggertsson, S. Kristmundsdottir, D. Beyter, H. Jonsson, A. Skuladottir, M. T. Hardarson, D. F. Gudbjartsson, K. Stefansson, B. V. Halldorsson, and P. Melsted, "GraphTyper2 enables population-scale genotyping of structural variation using pangenome graphs," vol. 10, no. 1, pp. 1–8. [Online]. Available: https://doi.org/10.1038/s41467-019-13341-9

[69] J. Huddleston, M. J. P. Chaisson, K. M. Steinberg, W. Warren, K. Hoekzema, D. Gordon, T. A. Graves-Lindsay, K. M. Munson, Z. N. Kronenberg, L. Vives, P. Peluso, M. Boitano, C.-S. Chin, J. Korlach, R. K. Wilson, and E. E. Eichler, "Discovery and genotyping of structural variation from long-read haploid genome sequence data," vol. 27, no. 5, pp. 677–685. [Online]. Available: https://genome.cshlp.org/content/27/5/677

[70] M. Kirsche, G. Prabhu, R. Sherman, B. Ni, A. Battle, S. Aganezov, and M. C. Schatz, "Jasmine and Iris: Population-scale structural variant comparison and analysis," vol. 20, no. 3, pp. 408–417. [Online]. Available: https://www.nature.com/articles/s41592-022-01753-3

[71] 1000 Genomes Project Consortium, A. Auton, L. D. Brooks, R. M. Durbin, E. P. Garrison, H. M. Kang, J. O. Korbel, J. L. Marchini, S. McCarthy, G. A. McVean, and G. R. Abecasis, "A global reference for human genetic variation," vol. 526, no. 7571, pp. 68–74.

[72] S. Chen, L. C. Francioli, J. K. Goodrich, R. L. Collins, M. Kanai, Q. Wang, J. Alföldi, N. A. Watts, C. Vittal, L. D. Gauthier, T. Poterba, M. W. Wilson, Y. Tarasova, W. Phu, M. T. Yohannes, Z. Koenig, Y. Farjoun, E. Banks, S. Donnelly, S. Gabriel, N. Gupta, S. Ferriera, C. Tolonen, S. Novod, L. Bergelson, D. Roazen, V. Ruano-Rubio, M. Covarrubias, C. Llanwarne, N. Petrillo, G. Wade, T. Jeandet, R. Munshi, K. Tibbetts, g. P. Consortium, A. O'Donnell-Luria, M. Solomonson, C. Seed, A. R. Martin, M. E. Talkowski, H. L. Rehm, M. J. Daly, G. Tiao, B. M. Neale, D. G. MacArthur, and K. J. Karczewski. A genome-wide mutational constraint map quantified from variation in 76,156 human genomes. [Online]. Available: https://www.biorxiv.org/content/10.1101/2022.03.20.485034v2

[73] Z. Iqbal, I. Turner, and G. McVean, "High-throughput microbial population genomics using the Cortex variation assembler," vol. 29, no. 2, pp. 275–276. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3546798/

[74] B. Solomon and C. Kingsford, "Large-Scale Search of Transcriptomic Read Sets with Sequence Bloom Trees," p. 017087. [Online]. Available: http://biorxiv.org/content/early/2015/03/26/017087.abstract

[75] M. Karasikov, H. Mustafa, D. Danciu, M. Zimmermann, C. Barber, G. Rätsch, and A. Kahles. MetaGraph: Indexing and Analysing Nucleotide Archives at Petabase-scale. [Online]. Available: https://www.biorxiv.org/content/10.1101/2020.10.01.322164v2

[76] C. Marchet, Z. Iqbal, D. Gautheret, M. Salson, and R. Chikhi, "REINDEER: Efficient indexing of k-mer presence and abundance in sequencing datasets," p. 2020.03.29.014159.

[77] M. Kokot, M. Dlugosz, and S. Deorowicz, "KMC 3: Counting and manipulating k-mer statistics," vol. 33, no. 17, pp. 2759–2761. [Online]. Available: http://sun.aei.polsl.pl/REFRESH/kmc.

[78] I. Turner, K. V. Garimella, Z. Iqbal, and G. McVean, "Integrating long-range connectivity information into de Bruijn graphs," vol. 34, no. 15, pp. 2556–2565.

[79] N. Li and M. Stephens, "Modeling Linkage Disequilibrium and Identifying Recombination Hotspots Using Single-Nucleotide Polymorphism Data," vol. 165, no. 4, pp. 2213–2233. [Online]. Available: https://doi.org/10.1093/genetics/165.4.2213

[80] B. L. Browning, X. Tian, Y. Zhou, and S. R. Browning, "Fast two-stage phasing of large-scale sequence data," vol. 108, no. 10, pp. 1880–1890. [Online]. Available: https://www.cell.com/ajhg/abstract/S0002-9297(21)00304-9

[81] SciPy documentation — SciPy v1.11.3 Manual. [Online]. Available: https://docs.scipy.org/doc/scipy/index.html

[82] I. Letunic and P. Bork, "Interactive Tree Of Life (iTOL) v5: An online tool for phylogenetic tree display and annotation," vol. 49, no. W1, pp. W293–W296. [Online]. Available: https://doi.org/10.1093/nar/gkab301

[83] B. L. Browning, Y. Zhou, and S. R. Browning, "A One-Penny Imputed Genome from Next-Generation Reference Panels," vol. 103, no. 3, pp. 338–348. [Online]. Available: https://www.cell.com/ajhg/abstract/S0002-9297(18)30242-8

[84] P. Ebert, P. A. Audano, Q. Zhu, B. Rodriguez-Martin, D. Porubsky, M. J. Bonder, A. Sulovari, J. Ebler, W. Zhou, R. Serra Mari, F. Yilmaz, X. Zhao, P. Hsieh, J. Lee, S. Kumar, J. Lin, T. Rausch, Y. Chen, J. Ren, M. Santamarina, W. Höps, H. Ashraf, N. T. Chuang, X. Yang, K. M. Munson, A. P. Lewis, S. Fairley, L. J. Tallon, W. E. Clarke, A. O. Basile, M. Byrska-Bishop, A. Corvelo, U. S. Evani, T.-Y. Lu, M. J. P. Chaisson, J. Chen, C. Li, H. Brand, A. M. Wenger, M. Ghareghani, W. T. Harvey, B. Raeder, P. Hasenfeld, A. A. Regier, H. J. Abel, I. M. Hall, P. Flicek, O. Stegle, M. B. Gerstein, J. M. C. Tubio, Z. Mu, Y. I. Li, X. Shi, A. R. Hastie, K. Ye, Z. Chong, A. D. Sanders, M. C. Zody, M. E. Talkowski, R. E. Mills, S. E. Devine, C. Lee, J. O. Korbel, T. Marschall, and E. E. Eichler, "Haplotype-resolved diverse human genomes and integrated analysis of structural variation," vol. 372, no. 6537, p. eabf7117. [Online]. Available: https://www.science.org/doi/10.1126/science.abf7117

[85] W.-W. Liao, M. Asri, J. Ebler, D. Doerr, M. Haukness, G. Hickey, S. Lu, J. K. Lucas, J. Monlong, H. J. Abel, S. Buonaiuto, X. H. Chang, H. Cheng, J. Chu, V. Colonna, J. M. Eizenga, X. Feng, C. Fischer, R. S. Fulton, S. Garg, C. Groza, A. Guarracino, W. T. Harvey, S. Heumos, K. Howe, M. Jain, T.-Y. Lu, C. Markello, F. J. Martin, M. W. Mitchell, K. M. Munson, M. N. Mwaniki, A. M. Novak, H. E. Olsen, T. Pesout, D. Porubsky, P. Prins, J. A. Sibbesen, J. Sirén, C. Tomlinson, F. Villani, M. R. Vollger, L. L. Antonacci-Fulton, G. Baid, C. A. Baker, A. Belyaeva, K. Billis, A. Carroll, P.-C. Chang, S. Cody, D. E. Cook, R. M. Cook-Deegan, O. E. Cornejo, M. Diekhans, P. Ebert, S. Fairley, O. Fedrigo, A. L. Felsenfeld, G. Formenti, A. Frankish, Y. Gao, N. A. Garrison, C. G. Giron, R. E. Green, L. Haggerty, K. Hoekzema, T. Hourlier, H. P. Ji, E. E. Kenny, B. A. Koenig, A. Kolesnikov, J. O. Korbel, J. Kordosky, S. Koren, H. Lee, A. P. Lewis, H. Magalhães, S. Marco-Sola, P. Marijon, A. McCartney, J. McDaniel, J. Mountcastle, M. Nattestad, S. Nurk, N. D. Olson, A. B. Popejoy, D. Puiu, M. Rautiainen, A. A. Regier, A. Rhie, S. Sacco, A. D. Sanders, V. A. Schneider, B. I. Schultz, K. Shafin, M. W. Smith, H. J. Sofia, A. N. Abou Tayoun, F. Thibaud-Nissen, F. F. Tricomi, J. Wagner, B. Walenz, J. M. D. Wood, A. V. Zimin, G. Bourque, M. J. P. Chaisson, P. Flicek, A. M. Phillippy, J. M. Zook, E. E. Eichler, D. Haussler, T. Wang, E. D. Jarvis, K. H. Miga, E. Garrison, T. Marschall, I. M. Hall, H. Li, and B. Paten, "A draft human pangenome reference," vol. 617, no. 7960, pp. 312–324. [Online]. Available: https://www.nature.com/articles/s41586-023-05896-x

[86] P. Danecek, J. K. Bonfield, J. Liddle, J. Marshall, V. Ohan, M. O. Pollard, A. Whitwham, T. Keane, S. A. McCarthy, R. M. Davies, and H. Li, "Twelve years of SAMtools and BCFtools," vol. 10, no. 2, p. giab008. [Online]. Available: https://doi.org/10.1093/gigascience/giab008

[87] M. Byrska-Bishop, U. S. Evani, X. Zhao, A. O. Basile, H. J. Abel, A. A. Regier, A. Corvelo, W. E. Clarke, R. Musunuri, K. Nagulapalli, S. Fairley, A. Runnels, L. Winterkorn, E. Lowy, E. E. Eichler, J. O. Korbel, C. Lee, T. Marschall, S. E. Devine, W. T. Harvey, W. Zhou, R. E. Mills, T. Rausch, S. Kumar, C. Alkan, F. Hormozdiari, Z. Chong, Y. Chen, X. Yang, J. Lin, M. B. Gerstein, Y. Kai, Q. Zhu, F. Yilmaz, C. Xiao, P. Flicek, S. Germer, H. Brand, I. M. Hall, M. E. Talkowski, G. Narzisi, and M. C. Zody, "High-coverage whole-genome sequencing of the expanded 1000 Genomes Project cohort including 602 trios," vol. 185, no. 18, pp. 3426–3440.e19. [Online]. Available: https://www.cell.com/cell/abstract/S0092-8674(22)00991-6

[88] A. Bergström, S. A. McCarthy, R. Hui, M. A. Almarri, Q. Ayub, P. Danecek, Y. Chen, S. Felkel, P. Hallast, J. Kamm, H. Blanché, J.-F. Deleuze, H. Cann, S. Mallick, D. Reich, M. S. Sandhu, P. Skoglund, A. Scally, Y. Xue, R. Durbin, and C. Tyler-Smith, "Insights into human genetic variation and population history from 929 diverse genomes," vol. 367, no. 6484, p. eaay5012. [Online]. Available: https://www.science.org/doi/10.1126/science.aay5012

[89] S. Mallick, H. Li, M. Lipson, I. Mathieson, M. Gymrek, F. Racimo, M. Zhao, N. Chennagiri, S. Nordenfelt, A. Tandon, P. Skoglund, I. Lazaridis, S. Sankararaman, Q. Fu, N. Rohland, G. Renaud, Y. Erlich, T. Willems, C. Gallo, J. P. Spence, Y. S. Song, G. Poletti, F. Balloux, I. G. Romero, A. R. Jha, D. M. Behar, C. M. Bravi, C. Capelli, T. Hervig, A. Moreno-Estrada, O. L. Posukh, E. Balanovska, O. Balanovsky, S. Karachanak-Yankova, H. Sahakyan, D. Toncheva, L. Yepiskoposyan, C. Tyler-Smith, Y. Xue, M. S. Abdullah, A. Ruiz-Linares, C. M. Beall, A. Di Rienzo, C. Jeong, E. B. Starikovskaya, E. Metspalu, J. Parik, R. Villems, B. M. Henn, U. Hodoglugil, R. Mahley, A. Sajantila, G. Stamatoyannopoulos, J. T. S. Wee, R. Khusainova, E. Khusnutdinova, S. Litvinov, G. Ayodo, D. Comas, M. F. Hammer, T. Kivisild, W. Klitz, C. A. Winkler, D. Labuda, M. Bamshad, L. B. Jorde, S. A. Tishkoff, W. S. Watkins, M. Metspalu, S. Dryomov, R. Sukernik, L. Singh, K. Thangaraj, S. Pääbo, J. Kelso, N. Patterson, and D. Reich, "The Simons Genome Diversity Project: 300 genomes from 142 diverse populations," vol. 538, no. 7624, pp. 201–206. [Online]. Available: https://www.nature.com/articles/nature18964

[90] M. Martin, M. Patterson, S. Garg, S. O. Fischer, N. Pisanti, G. W. Klau, A. Schöenhuth, and T. Marschall. WhatsHap: Fast and accurate read-based phasing. [Online]. Available: https://www.biorxiv.org/content/10.1101/085050v2

[91] T. Yun, H. Li, P.-C. Chang, M. F. Lin, A. Carroll, and C. Y. McLean, "Accurate, scalable cohort variant calls using DeepVariant and GLnexus," vol. 36, no. 24, pp. 5582–5589. [Online]. Available: https://doi.org/10.1093/bioinformatics/btaa1081

[92] J. M. Zook, N. F. Hansen, N. D. Olson, L. M. Chapman, J. C. Mullikin, C. Xiao, S. Sherry, S. Koren, A. M. Phillippy, P. C. Boutros, S. M. E. Sahraeian, V. Huang, A. Rouette, N. Alexander, C. E. Mason, I. Hajirasouliha, C. Ricketts, J. Lee, R. Tearle, I. T. Fiddes, A. M. Barrio, J. Wala, A. Carroll, N. Ghaffari, O. L. Rodriguez, A. Bashir, S. Jackman, J. J. Farrell, A. M. Wenger, C. Alkan, A. Soylev, M. C. Schatz, S. Garg, G. Church, T. Marschall, K. Chen, X. Fan, A. C. English, J. A. Rosenfeld, W. Zhou, R. E. Mills, J. M. Sage, J. R. Davis, M. D. Kaiser, J. S. Oliver, A. P. Catalano, M. J. Chaisson, N. Spies, F. J. Sedlazeck, M. Salit, and the Genome in a Bottle Consortium, "A robust benchmark for germline structural variant detection." [Online]. Available: http://biorxiv.org/lookup/doi/10.1101/664623

[93] A. C. English, V. K. Menon, R. Gibbs, G. A. Metcalf, and F. J. Sedlazeck. Truvari: Refined Structural Variant Comparison Preserves Allelic Diversity. [Online]. Available: https://www.biorxiv.org/content/10.1101/2022.02.21.481353v1

[94] V. Geoffroy, Y. Herenger, A. Kress, C. Stoetzel, A. Piton, H. Dollfus, and J. Muller, "AnnotSV: An integrated tool for structural variations annotation," vol. 34, no. 20, pp. 3572–3574. [Online]. Available: https://doi.org/10.1093/bioinformatics/bty304

[95] I. Pokrovac and Z. Pezer, "Recent advances and current challenges in population genomics of structural variation in animals and plants," vol. 13. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fgene.2022.1060898

[96] R. Leinonen, H. Sugawara, and M. Shumway, "The Sequence Read Archive," vol. 39, pp. D19–D21. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3013647/

[97] K. V. Garimella, Z. Iqbal, M. A. Krause, S. Campino, M. Kekre, E. Drury, D. Kwiatkowski, J. M. Sá, T. E. Wellems, and G. McVean, "Detection of simple and complex de novo mutations with multiple reference sequences," vol. 30, no. 8, pp. 1154–1169. [Online]. Available: https://genome.cshlp.org/content/30/8/1154.full

[98] C. Chiang, A. J. Scott, J. R. Davis, E. K. Tsang, X. Li, Y. Kim, T. Hadzic, F. N. Damani, L. Ganel, S. B. Montgomery, A. Battle, D. F. Conrad, and I. M. Hall, "The impact of structural variation on human gene expression," vol. 49, no. 5, pp. 692–699. [Online]. Available: https://www.nature.com/articles/ng.3834

[99] N. K. Kadri, G. Sahana, C. Charlier, T. Iso-Touru, B. Guldbrandtsen, L. Karim, U. S. Nielsen, F. Panitz, G. P. Aamand, N. Schulman, M. Georges, J. Vilkki, M. S. Lund, and T. Druet, "A 660-Kb Deletion with Antagonistic Effects on Fertility and Milk Production Segregates at High Frequency in Nordic

Red Cattle: Additional Evidence for the Common Occurrence of Balancing Selection in Livestock," vol. 10, no. 1, p. e1004049. [Online]. Available: https://journals.plos.org/plosgenetics/article?id=10.1371/journal.pgen.1004049

[100] Y. Hou, G. E. Liu, D. M. Bickhart, L. K. Matukumalli, C. Li, J. Song, L. C. Gasbarre, C. P. Van Tassell, and T. S. Sonstegard, "Genomic regions showing copy number variations associate with resistance or susceptibility to gastrointestinal nematodes in Angus cattle," vol. 12, no. 1, pp. 81–92.

[101] M. Boussaha, D. Esquerré, J. Barbieri, A. Djari, A. Pinton, R. Letaief, G. Salin, F. Escudié, A. Roulet, S. Fritz, F. Samson, C. Grohs, M. Bernard, C. Klopp, D. Boichard, and D. Rocha, "Genome-Wide Study of Structural Variants in Bovine Holstein, Montbéliarde and Normande Dairy Breeds," vol. 10, no. 8, p. e0135931. [Online]. Available: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0135931

[102] Y.-L. Lee, M. Bosse, H. Takeda, G. C. M. Moreira, L. Karim, T. Druet, C. Oget-Ebrad, W. Coppieters, R. F. Veerkamp, M. A. M. Groenen, M. Georges, A. C. Bouwman, and C. Charlier, "High-resolution structural variants catalogue in a large-scale whole genome sequenced bovine family cohort data," vol. 24, no. 1, p. 225. [Online]. Available: https://doi.org/10.1186/s12864-023-09259-8

[103] L. Chen, A. J. Chamberlain, C. M. Reich, H. D. Daetwyler, and B. J. Hayes, "Detection and validation of structural variations in bovine whole-genome sequence data," vol. 49, no. 1, p. 13. [Online]. Available: https://doi.org/10.1186/s12711-017-0286-5

[104] A. S. Leonard, D. Crysnanto, Z.-H. Fang, M. P. Heaton, B. L. Vander Ley, C. Herrera, H. Bollwein, D. M. Bickhart, K. L. Kuhn, T. P. L. Smith, B. D. Rosen, and H. Pausch, "Structural variant-based pangenome construction has low sensitivity to variability of haplotype-resolved bovine assemblies," vol. 13, no. 1, p. 3012. [Online]. Available: https://www.nature.com/articles/s41467-022-30680-2

[105] W. Y. Low, R. Tearle, R. Liu, S. Koren, A. Rhie, D. M. Bickhart, B. D. Rosen, Z. N. Kronenberg, S. B. Kingan, E. Tseng, F. Thibaud-Nissen, F. J. Martin, K. Billis, J. Ghurye, A. R. Hastie, J. Lee, A. W. Pang, M. P. Heaton, A. M. Phillippy, S. Hiendleder, T. P. Smith, and J. L. Williams, "Haplotype-resolved genomes provide insights into structural variation and gene content in Angus and Brahman cattle," vol. 11, no. 1, pp. 1–14. [Online]. Available: https://doi.org/10.1038/s41467-020-15848-y

[106] D. Heller and M. Vingron, "SVIM-asm: Structural variant detection from haploid and diploid genome assemblies," vol. 36, no. 22-23, pp. 5519–5521. [Online]. Available: https://doi.org/10.1093/bioinformatics/btaa1034

[107] "PacificBiosciences/pbmm2," PacBio. [Online]. Available: https://github.com/PacificBiosciences/pbmm2

[108] D. C. Jeffares, C. Jolly, M. Hoti, D. Speed, L. Shaw, C. Rallis, F. Balloux, C. Dessimoz, J. Bähler, and F. J. Sedlazeck, "Transient structural variations have strong effects on quantitative traits and reproductive isolation in fission yeast," vol. 8, no. 1, pp. 1–11. [Online]. Available: www.nature.com/naturecommunications

[109] C. T. Brown and L. Irber, "Sourmash: A library for MinHash sketching of DNA," vol. 1, no. 5, p. 27. [Online]. Available: https://joss.theoj.org/papers/10.21105/joss.00027

[110] W. McLaren, L. Gil, S. E. Hunt, H. S. Riat, G. R. S. Ritchie, A. Thormann, P. Flicek, and F. Cunningham, "The Ensembl Variant Effect Predictor," vol. 17, no. 1, p. 122. [Online]. Available: https://doi.org/10.1186/s13059-016-0974-4

[111] Calculated consequences. [Online]. Available: https://www.ensembl.org/info/genome/variation/prediction/predicted_data.html

[112] M. Kirsche, G. Prabhu, R. Sherman, B. Ni, A. Battle, S. Aganezov, and M. C. Schatz, "Jasmine and Iris: Population-scale structural variant comparison and analysis," vol. 20, no. 3, pp. 408–417. [Online]. Available: https://www.nature.com/articles/s41592-022-01753-3

[113] A. Clop, O. Vidal, and M. Amills, "Copy number variation in the genomes of domestic animals," vol. 43, no. 5, pp. 503–517.