CSCI 4192 Directed Studies

Project Report

Supervisor: Dr. Travis Gagie

Prepared By

Nathaniel Brown B00766703

Le Wang B00761974

Yansong Li B00755354

Dec 16, 2020

Table of Contents

1. Introduction

Bioinformatics and specifically the field of genomics have made great strides in only the past 20 years. Following the Human Genome Project, a reference genome was constructed, giving an efficient alternative to genome assembly. With read-alignment and an indexed reference genome using tools using the FM-Index, your genome can be recovered by matching sections of your sequence against the reference using matching queries.

However, the common approaches do not leverage using multiple genomes in the reference well, becoming highly inefficient. In motivation of alternative approaches, we cover reference bias and other concerns of using only a single reference genome. Despite the difficulties, technology such as variations graphs and the R-Index have allowed us to query multiple genome references, allowing genome recovery using more variation.

Having surveyed the current topics in genomics, we explore what number of genomes ought to be included when indexing for sequencing. Using results from the R-Index, matching statistics were computed using PHONI technology, allowing us to compute the RLZ Parse size which denotes the compressibility of a pattern with respect to a reference. In our experiment, it relates how much new information (in terms of variation) is added with each additional copy. We show the results of using various numbers of copies of chromsome-19 and conclude that using at least 64 genomes maximizes early returns. However, future work should explore the relationship between value and including more reference genomes further.

2. Reference Genome

The genome of every individual organism in this world has a specific sequence composition, but there are also similarities between different species or within the same species. Geneticists have completed a haplotype single human genome sequencing during the Human Genome Project in 2003. Furthermore, geneticists have launched the Earth BioGenome Project (EBP) to sequence the genomes of the rest of the eukaryotic species on the earth in 2018 (Ballouz, 2019).

After the Human Genome Project, geneticists have sequenced the first human reference genome; however, it was just an arbitrary reference point since it was only sequenced from a person's genome. Such a reference genome can cause reference bias, so geneticists are currently working on assembling more representative reference genomes with a larger sample, such as the 1000 Genome Project. After many individual donors' DNA is sequenced, geneticists will be able to build an idealized complete set of genes from an artificial individual organism, which does not represent any single individual organism but the entire species. For instance, the GRCm39 released by the Genome Reference Consortium in 2019 is a reference genome like that (U.S. National Library of Medicine).

The reference genome is commonly used as a guide for constructing new genome sequences, helping to solve the read alignment problem. If we consider genome alignment akin to a puzzle, then the large number of sequencing reads (using sequence by synthesis or other methods) are the pieces of this puzzle. Then, the reference genome would be the final picture of the puzzle, helping to align the reads so that we can complete the picture. With help from the reference genome and read alignment approach, indexing a genome is faster and easier then assembling from scratch (Langmead, 2016).

More specifically, reference genomes can help geneticists solve many real-world problems. For example, they can hope to study and identify rare genetic diseases in children, comparing the difference between the human reference and the children's genomes. For the same reason, scientists can also study ancient humans' genomes, tumors, bacteria, and how genomes work. The reference genome plays an essential guide role in all these problems (Langmead, 2016). However, better alignment and using multiple references is key in identifying differences that the single reference will miss. To motivate the reference genome, we first survey current methods of recovering a genome from a sample.

3. Recovering Genomes

3.1 Sequencing by synthesis

In preparation of recovering genomes, geneticists generally use a method called

sequencing by synthesis to generate sequencing reads. Using a blood sample, we extract the double-stranded DNA, separating it into short snippets in the process, creating many copies. Next, we spilt the double-stranded snippets into the single-stranded template, so we can deposit them on a slide (a flat surface) and scatter them randomly across it. After that, we use DNA polymerase to match the single-stranded template with their complementary bases. However, the newly generated complementary bases are limited by terminators, which means they only have a height of 1 at the beginning. Meanwhile, we take a photo from the top of the slide and the terminator will glow a particular color that corresponds to the base. Therefore, the photo that we take will only have the current glowing terminators. Then we remove the terminators and add new complementary bases on top of the original for each template and repeat the photo-taking process. As a result, we will get a series of photos and each glowing terminator with the same position represents the complementary bases on our original template strand.

However, when we release the polymerase, an error can occur in which some bases are not terminated. When a non-terminated base is added, the polymerase will naturally place another base on top of it (likely to be terminated). We are then ahead of schedule, and our photograph will contain some light that is incorrect from these out of sync bases. For this reason, reads generated using sequencing by synthesis are generally kept short. Alternative methods produce longer reads but are similarly prone to error. Therefore, we focus on sequencing by synthesis for now, where we use these photos to generate sequencing reads for all templates (Langmead, 2016). Once the geneticists obtain sequencing reads (using sequencing by synthesis, for example), there are two types of approaches used to recover a completed single genome from sequencing reads. The first approach is read alignment and the second one is assembly.

3.2 Read alignment

For the read alignment approach, our primary method is using a reference genome as a guide to align reads, since genomes from the same species are very similar (99.8%). Specifically, we take one sequencing read then find a place in the reference genome that

matches the read sufficiently (with few errors). Next, we repeat this process to match all sequencing reads in our dataset with the reference genome, a difficult task since the length of the genome is more than three billion bases long (Langmead, 2016).

Indexing is a great offline algorithm for rapidly finding specific content, such as in a book, where we associate key terms with their respective page numbers of occurrences. Therefore, if we preprocess the reference genome by marking the indexes where sequencing reads occur, then we can recover the target genome faster. We treat both the sequencing read and reference genome as strings, helping us decrease the difficulty of genome indexing by relating the problem to well-studied approaches. Once we have an efficient index for the reference, we can easily query for certain reads using two types of matching approaches (Langmead, 2016).

An exact matching approach is ideal when there is a perfect match of the sequencing read inside the reference genome, which we can implement using the Boyer-Moore string matching algorithm. However, exact matching is often not sufficient, due to sequencing errors and natural variation between the genomes which avoid exact matches. For this trickier scenario, we use approximate matching to align reads. This approach is not as greedy as exacting matching; in approximate matching, our goal is just to find a substring of the reference genome that approximately matches the sequencing read with an acceptable edit distance (mismatched bases, insertions, deletions) between them.

Specifically, we split the sequencing read into k+1 subreads where k is the maximum allowed edit distance. According to the pigeonhole principle, we can find at least one subread that exactly matches somewhere in the reference genome if this read can be matched with up to k edit distance. Next, we extend the exactly matched subread and check whether its neighbor bases are matched with the neighbors of the matched spot in the reference genome. If we can do this without exceeding our edit distance, then we consider that is a match, choosing matches with the lowest edit distance as the final aligned position. Hence, the substance of approximate matching is in converting exact matching to approximate matching using the pigeonhole principle (Langmead, 2016).

3.3 Assembly

The assembly approach is also known as De Novo Shotgun Assembly, where De Novo means working from scratch and shotgun means the snippet reads are coming randomly from all over the genome, without indexing the reference. Therefore, the objective of this assembly method follows from its name; we reconstruct a DNA sequence from scratch using random snippet reads, overlapping them with an acceptable error rate. However, this is a very difficult task to accomplish with current technical constraints. Specifically, the length of the genome fragment that can be read at one time with the existing technology is too short, and shorter snippets will lead to a higher error rate due to over collapsing and other problems. Therefore, the DNA sequence assembly challenge has explicit objectives and method, but is constrained by current technology (Langmead, 2016).

However, constantly developing and cutting-edge technology has given us aid, allowing us to develop a solution which grows closer to the ideal scenario with each new advancement. There are two types of graphs which can help with this challenge: overlap graphs, and De Bruijn graphs. Overlap graphs are based on the first and the second law of assembly, which are “if a suffix of read A is like a prefix of read B, then A and B might overlap in the genome,” and “more coverage leads to more and longer overlaps”. Overlap graphs are directed graphs that take the DNA sequence reads as nodes and take overlaps between the suffix of one DNA sequence read and prefix of another as edges. Once we have constructed the overlap graph of a set of DNA sequence reads, then we can assemble the genome by finding the shortest common superstring of it.

Nevertheless, the shortest common superstring does not necessarily recover the pieces of the genome due to over collapsing; therefore, scientists also use another method, the De Bruijn graph. De Bruijn graphs allow for more than one edge between two nodes, using a Eulerian Walk for DNA sequence assembly, eliminating the problem of over collapsing. However, the number of Eulerian Walks is unstable; a De Bruijn graph could have multiple Eulerian Walks or none at all. Although both graphs have flaws, they are still valuable resources for DNA sequence assembly (Langmead, 2016).

Even though scientists do not have ideal solutions to this challenge, there are some partial solutions that have been developed. The reason repeating subsequences is so annoying is that the sequence fragments are too short. For instance, if a puzzle has been covered by many featureless pieces, then it becomes very hard to put it back together, especially in assembly where we don't know the full picture. However, if the number of featureless pieces decreases (by increasing the size), then the difficult in placing them is reduced as well. Hence, one solution to this challenge is producing larger DNA sequence snippets to reduce the effect of repetitive subsequences. When the featureless pieces are concatenated with some unique features, the puzzle will be easier to put together. Further, another solution is to concatenate non-repetitive subsequence before and after the repetitive DNA reads. Both solutions are designed to address part of this challenge, but obviously, they have not been fully implemented. Thus, recovering a genome by alignment is much preferred to assembly, emphasizing the importance of the reference genome and specifically our ability to index it. (Langmead, 2016).

4. Indexing a Single Reference Genome

The dominating approach when indexing a single reference genome is the FM-Index, which underlies popular tools such as Bowtie and BWA (Langmead, 2020). It can compress a full genome while still allowing efficient substring queries. Armed with an indexed reference genome, we can easily sequence and align reads using the pigeonhole principle, seeding and extending, to recover a genome sequence.

The FM-Index's core approach is using the Burrows-Wheeler transform (BWT). This consists of exhausting all rotations (or shifts) of the text to index and sorting lexicographically to recover the Burrows-Wheeler matrix (using $ or another character outside of the alphabet to signal the end of the text). The last column of this matrix is the burrows wheeler transform; a BTW element in the row is the preceding character of the rest of the row, such that the rest of the sequence is the right context of this element.

Interestingly, we see that each character in the first column and each character in the last column (BWT) share the same rank, called the LF-Mapping, where the rank is the numbered occurrence of when that specific character appears. In other words, the third A in the first column, as it appears in the text, which we denote A_3, can be found by finding the third A in the last column. This property follows from the lexicographical sorting alongside the definition of right context.

With the LF-Mapping, we can store only the last (BWT) and first columns of the Burrows-Wheeler Matrix. This is because we can recover by reversal the full text. Starting at the first element of the first column (terminating character), we find the preceding character in the same row, but in the last column of the matrix. By querying the rank of this character, we can identify where this character is in the first column. Then, we simply repeat, until we have reversed the transformation using only the two columns. This makes the procedure highly compressible, requiring only these two columns, and further since sorting by lexicographical rank gives many repetitions in which can be compressed further.

However, to efficiently perform these necessary rank queries the FM-Index can leverage bitvectors and wavelet trees to make these rank queries more efficient. Using our alphabet, we can use access, rank, and select queries to recover the characters more efficiently by representing the bitvector encoding in a partitioned tree hierarchy which limits calls. Although this primarily speeds up access queries (find character at position j), we rely on access to in calculating rank queries as well.

Armed with efficient rank queries, the FM-Index is highly compressible, sped up by wavelet tree queries. The index itself primarily supports count and locate queries with respect to a pattern P, where locate returns the offsets where P matches in the text, and count returns the number of these offsets. The core of performing these queries is starting with the shortest suffix of our pattern and matching successively longer patterns. Given A, we find the rows starting with A, and then through LF-Mapping, if our next suffix is BA, find where a B preceded this A, until we have matched the whole pattern, or we fail. To return the indices by locate queries, we store a sampled suffix array holding the actual indices (storing the whole

array would be costly), leveraging that if our match has not value in the suffix array, we can trace using the LF-Mapping to find the how far back it is from the next sampled index.

Thus, the LF-Mapping allows us to efficiently index a reference genome and query it given aligned reads. Indeed, the FM-Index is used by bioinformaticians around the world to recover genomes using read-alignment (Gagie, 2020a). However, what if we wish to include multiple genomes in the reference? The FM-Index grows linearly with respect to the additional text. Indeed, additional sequences are treated as linear additions, and indexing multiple genomes quickly slows down query time. To see how this is problematic, we investigate the benefits of including multiple genomes over a single reference genome.

5. Benefits of using Multiple References

A single reference genome is not particularly representative of the full genetic variation of a species. Since we want to catalogue as much human genetic variation as possible to ensure our assembly results are accurate, references with which use multiple genomes are of benefit. Also, some of the human genes are in hypervariable regions, where it highly varies from person to person, therefore, making use of a single reference genome problematic. We find similar motivation when considering allele-specific genetics.

A gene has both paternal and maternal copy; suppose we want to find whether the gene is more highly expressed from the paternal copy of the maternal copy, or equally expressed across both. With only one reference genome, it is more likely to have many reference alleles on one copy but few on the other. The read-aligner must compare the number of reference alleles and non-reference alleles on both copies in order to align the reads. This becomes more difficult as it needs to overcome the penalty caused by mismatches with every non-reference allele. Thus, it may result in a strong bias towards one of the copies.

To further illustrate issues caused by using only one reference genome, we explore an example relating to rare-genetic occurrences. A genetic mutation had caused a boy's abnormalities: a flattened face, cognitive delays, cleft palate, stubby thumbs, and a host of other skeletal malformations. The geneticists believed that he had a rare disease called Baratela Scott,

so they followed the usual procedure and checked the spelling of the boy's DNA sequences to the reference human genome. However, the boy's sequence showed no sign of the mutation in the gene known to cause Baratela Scott, called XYLT1. In fact, many pieces of the boy's genomes were not in the reference genome at all, meaning there was no way to check them for disease-causing misspellings.

The reference genome's shortcomings are rooted in its history. In 1997, Buffalo's newspaper placed an ad seeking volunteers to donate blood from which experts would sequence DNA. About 70 percent of the reference genome came from an anonymous man designated RP11, and the rest from a few other volunteers. It was not a perfectly healthy genome, as it had at least 3556 variants that increase the risk of diseases, including type 1 diabetes and hypertension. Moreover, its ethnic populations were almost all European - German, Irish, Polish, which was very European-biased, which means that genetic code of African ethnicity would produce more differences from the reference than European origin. Indeed, Biologist Steven Salzberg of John Hopkins University sequenced the genomes of 900 African Americans and found many of their pieces, nearly 296,485,284 base pairs, were missing from the reference genome. This example, among the other presented, motivates the need to use multiple reference genomes.

6. Difficulties of Indexing Multiple Genomes

Indexing a single genome is a relatively achievable and economical task in recent years; it can be accomplished within a day and with less than $1000 (Gagie, 2020a). However, such a reference genome cannot be representative of its entire species because of the diversity between different individual organisms. Therefore, the current task for geneticists is indexing multiple reference genomes, akin to a pangenome. Nevertheless, indexing multiple genomes simultaneously is a cutting-edge topic in genomics, and it still has many difficulties that need to be conquered. As we can recall, the popular FM-Index does not scale well when indexing multiple genomes.

In general, the current difficulties are either indexing being dramatically slow, costing a

huge amount of random-access memory (RAM), or receiving latency for online indexing. For slow-indexing problem, the current alignment tools such as Bowtie and Burrows-Wheeler Aligner (BWA) are designed to index linearly, which means they badly scale when indexing many genomes simultaneously. We can make data structures like the suffix array sample and the range minimum query (RMQ) smaller, but the cost is making them much slower (Gagie, 2020a), which can lead to an increase in the time required for the overall process of genome indexing.

Besides, genomes are different from usual strings even though we treat them as such, because they are extremely long. For example, “a human chromosome 1 is a quarter of a billion base pairs” (Boucher, 2020), and if we want to store the position values for all base pairs, it will take 2GB of RAM in general (Boucher, 2020). Therefore, it requires 46 GB of RAM just for buffering the position value for one person, and we also need RAM for other data structures. As a result, the total amount of RAM to index multiple genomes simultaneously is unreachable for most of the computers in the current stage. However, there are current solutions related to indexing multiple genomes.

7. Current Solutions to Indexing Multiple Genomes

7.1 Hybrid Indexing

Since human genomes are very similar, genomic databases are highly repetitive. However, conventional indexes are based on FM-indexes (or similar), which do not handle repetitive structure well when applied to many genomes, quickly outgrow internal memory. Thus, hybrid indexing is an alternative option, it reduces the size of conventional indexes on highly repetitive texts while preserving most or all their functionality (Ferrada, 2014).

For pattern lengths and edit distances, we give them upper bounds, then pre-process the text with the lossless data compression algorithm LZ77 to obtain a filtered text. Then, we find all matches with a query in that filtered text, then use these positions and the LZ77 structure to find the original text. This significantly reduces query times.

According to LZ77, for each phrase T[i..j] in the parse of T:

1. i = j and T[i] is the first occurrence of that distinct character, then T[i] is encoded as itself
2. T[i..j] occurs in T[1...j-1] but T[i...j+1] does not occur in T[1...j], then T[i...j] is encoded as the pair (i', j-i+1), where i' is the starting point of the leftmost occurrence of T[i...j] in T.

Here is an example to briefly explain the process of encoding. Suppose T is

99-bottles-of-beer-on-the-wall

Then, the parse of T (with parentheses around phrases) is:

(9)(-)(b)(o)(t)(t)(l)(e)(s)(-)(o)(f)(-b)(e)(e)(r)(-o)(n)(-)(t)(h)(e)(-)(w)(a)(l)(l)

and T's encoding is:

9 (1, 1) -bot (6, 1) les (3, 1) (5, 1) f (3, 2) (9, 1) (9, 1)

where,

- 9 is encoded as 9. As it is the first occurrence of 9.
- 9 is encoded as (1, 1). As this 9 is not the first occurrence of 9, it should be in the second case of LZ77. This 9's index is 2, i = j = 2, so T [2] = 9 occurs in T[1...2-1] = T[1] = 9, but T[2...3] = 9- does not occur in T[1...2] = 99. Therefore this 9 is encoded as (i' = 1, j-i+1) = (1, 2-2+1) = (1, 1).
- -bot is encoded as -bot. As it is the first occurrence of -bot, encoded as itself.
- t is encoded as (6, 1). As t is not the first occurrence of t, the index of t is 7, i = j = 7, T[7] = t occurs in T[1...6] = 99-bot, but T[7...8] = tl does not occur in T[1...7] = 99-bott, it is the second case of LZ77. So it is encoded as the pair (6, 7-7+1) = (6, 1).

...

- f is encoded as f itself.
- -b is encoded as (3, 2). -b is not the first occurrence of -b, the index of -b is i = 14, j = 15. T[14...15] = -b occurs in T[1...14] = 99-bottles-of-, but T[14...16] = -be does not occur in T[1...15] = 99-bottles-of-b. So it is the second case of LZ77, and encoded as the pair (3,

15 - 14 + 1) = (3, 2).

...

Primary matches

Suppose M and K be the upper bounds on pattern lengths and edit distances respectively. $T_{M, K}$ be the text containing only those characters of T within distance M + K - 1of their nearest phrase boundary in the LZ77 parse. Moreover, if the characters in T are not adjacent, then they will be separated in $T_{M, K}$ by K + 1 copies of # characters.

Secondary matches

For each primary match T[m...n], we find each phrase T[i...j] whose T[i'...i'+ j - i] includes T[m...n], we record T[m', n'] as a secondary recurrence and recurse on it, where m' = i + m - i' and n' = i + 2m - i' - n.

7.2 Founder sequences

To develop a sufficiently small, efficiently queryable, but still descriptive representation of the haplotype sequences that retain the original contiguities, we use founder sequences. Founder sequence reconstruction finds a set of d founders such that the original m haplotypes can be mapped with a minimum number of crossovers to the founders, where crossover means a position where one needs to jump from one founder to another to continue matching the content of the haplotype in question (Tuukka, 2019).

Suppose the input is the set R = {R1, ..., Rm} of strings of length n, called recombinants. A set F = {F1, ..., Fd] of strings of length n is called a founder set of R if for each string Ri belongs to R, there exists a partition Pi of the segment [1, n] into disjoint subsegments such that, for each [a, b] belongs to Pi, the string Ri[a, b] is equal to Fj[a, b] for some j belongs to [1. d].

7.3 Variation graph

The variation graph (VG) represents genetic variation as a graph. The path in the graph is

a potential haplotype. VG uses GCSA2 as its text index. GCSA2 represents a k-mer index as a de Bruijn graph and compresses it structurally by merging redundant nodes. VG handles complex graph regions by indexing a simplified graph and uses a simpler graph structure to replace the complex regions (Sirén, 2019).

A graph G can be represented by G = (V, E), V is the nodes, and E is the edges. We assume that the edges are directed, such that (u, v) is an edge from node u to node v. The in-degree of node v is the number of incoming edges to v, and the out-degree is the number of outgoing edges from v. If P = v0...v|p|-1 be a string over the set of nodes V, then P is a path in graph G = (V, E) if (vi, vi+1) belongs to E for all 0 <= i <= |p|-1.

For example, the Graph BWT (GBWT) is a scalable implementation of the graph extension of the positional Burrows-Wheeler transform (PBWT), which can store the haplotypes as paths in the graph. Since chromosome-length phasing is often not available, there may be multiple paths for each haplotype, then the GBWT can both represents arbitrary collections of paths over graphs and is encoded locally with the graph. Suppose P0, ..., Pm-1 be paths in G = (V, E), we can interpret the paths as strings over alphabet V. There is no 0 in nodes V, because we use it as the end-marker (Sirén, 2019).

The GBWT supports the following variants of the basic FM-index queries: find, locate and extract. These queries should be understood as examples of what we can support. Since the GBWT is an FM-index of multiple texts, most algorithms using an FM-index can be adapted to use the GBWT. For example, let $P = v_0 \dots v_{|p|-1}$ be a path, then the reverse-path of P is $P' = v'_{|p-1|} \dots v'_0$. The dynamic GBWT is a representation of the GBWT optimized for index construction, with speed prioritized over size. The compressed GBWT balances query performance with index size, we use it when the set of haplotypes is fixed and for storing the index on disk.

7.4 R-Index

While the FM-Index makes use of the compressibility of the single genome, it does not scale well indexing many genomes, growing proportional to the size of the sequences (Gagie, 2020b). The R-Index is designed to make use of the repetitiveness between sequences,

growing by the number of distinct sequences rather than total size of the sequences. Although the FM-Index is sufficient for single reference cases, the R-Index can query faster when many genomes are indexed without growing linearly.

The efficiency gains shown in R-Index were based off existing work improving the FM-Index and similar indexing approaches, namely the RLFM-Index (Gagie, 2020b). By improving rank and predecessor queries and leveraging existing knowledge of existing positions in the text T in O(r) size-sampling (where r is number of runs in burrows wheeler transform). Using fully functional suffix trees and this optimized text searching approach, by its name, the R-Index is bounded in O(r) space "outperforms the alternatives by 1–2 orders of magnitude in time when locating the occurrences of a pattern, while being simultaneously smaller or nearly as small" (Gagie, 2020b). As well, it is order of magnitudes faster and smaller than the FM-Index on repetitive datasets.

A significant improvement owed to the R-Index is the ability to produce matching statistics. If we are receiving a genome pattern online by character and character, then "the maximum latency from the time we receive a character to the time we return the corresponding pair in the matching statistics, also grows linearly with the length of the pattern" (Boucher, 2020). However, using PHONI, which works from the R-Index, we can compute matching statistics online with low latency, as well as compute for long patterns such as the genome in parallel with reasonable RAM usage (Boucher, 2020). This efficient computation allows us to produce and utilize matching statistics for multiple indexing in real time cases as well as others.

8. Using RLZ Parses to measure Multiple Indexing Benefits

The efficient computation of matching statistics by indexing a multiple genome reference allows us to compute measures which describe the value of using multiple genomes. In particular, matching statistics can be used in Relative Lempel-Ziv (RLZ) compression of a genome with respect to the reference (Kuruppu, 2020). Specifically, the RLZ Parse, which is

based LZ77 compression parsing, can be computed (in variant RLZAP) using matching statistics (Cox, 2016). If we are concerned with how compressible a sequenced genome is with respect to an indexed reference, we can compute the RLZ parse sizes.

To elaborate, the RLZ compression itself can compresses strings with respect to a reference with fast random access, highly efficient at exploiting similarity between strings, such as in genomes (Cox, 2016). To do so, the RLZ parse is constructed with respect to the patterns and a chosen reference string, partitioning the pattern into the minimum number of substrings that occur in the reference (or are a single character not occurring in the reference). The RLZ parse size is the number of such substrings.

Then, given the matching statistics taken from the R-Index, we can compute how compressible a pattern genome is with respect to the reference genome. The RLZ Parse size tells us how well it can be matched, since lower sizes indicate that we were able to find more similarity (matches) between the reference and our pattern. Relating this to multiple genomes, we can inspect how compressible a pattern is with respect to multiple genome references. This measure details how much new information using different number of genomes in the reference brings, in that lower RLZ Parse sizes indicate better ability to match a pattern (and similarly reads of that pattern, when sequencing).

To compute the RLZ Parse sizes themself, we leverage the matching statistics length values, which we isolate and denote $L[0..m-1]$ where our pattern $P[0..m-1]$ similarly has length m. To elaborate on matching statistics, $L[i]$ denotes the maximum length of a substring $P[i..i + L[i]]$ that occurs in the reference. Given L from the matching statistics, we can easily compute the RLZ Parse size by iterating over the lengths:

- Skipping indices which follow a new match, since if $P[i..i + L[i]]$ has been matched, then the following $L[i]-1$ indices are derivatives of that match (otherwise L[i] was not maximal)
- Incrementing the size when we begin a new matched substring
- Incrementing the size when we find a character that has not occurred before (where $L[i] = 0$)

We note that, alongside the online capabilities of PHONI in generating matching statistics, we can easily transform this approach to be online as well, we are only ever concerned with the present length values. Thus, we have an easily implementable method to compute RLZ Parse sizes given matching statistics.

9. Growth of RLZ Parse Size with Multiple Indexing

Using the methodology and approach justified above, we experiment using matching statistics generated by PHONI. We restrict ourselves to chromosome-19 of the genome, computing the matching statistics from matching one chromosome with respect to an R-Indexed reference of variable numbers of chromosome-19. Particularly, we compute the RLZ Parse size given matching statistics generated from a single copy of chromosome-19 with respect to a reference of 1, 4, 16, 64, 256, and 1000 copies. The results are shown in Table 1.

Table 1: Shows the computed RLZ Parse sizes with respect to a single chromosome-19 copy and a reference of the specified number of chromosome-19 copies

Copies	1	4	16	64	256	1000
RLZ Parse Sizes	1980645	1940510	1903411	1865906	1793812	1526362

Of course, we see that we are able to better compress the pattern copy with respect to the reference copy when we allow for more copies of chromosome-19 in the reference. Visualizing this trend in Figure 1, we see a well-structured descent of the RLZ Parse size. Interestingly, we see a logarithmic descent for values less than 64, and a linear descent for values greater than 64. The resulting line of best fit around the points of linear descent reveals a slope of approximately -361.84.

From our result, we can see that the greatest return on value is using multiple copies in the range from 1-64, increasing logarithmically with each step. However, this is a relatively small part of the graph, we see even with a linear increase, we can still gain a lot in terms of compression, and therefore gaining of new information with respect to matching. Of course, the difficulty of indexing such high numbers of copies must be weighed.

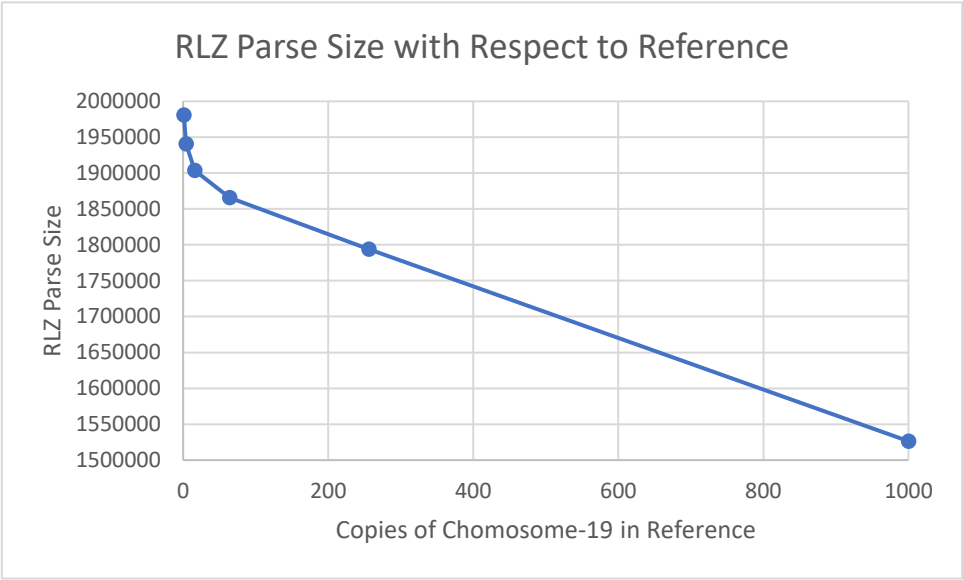

Figure 1: We graph the trend of the RLZ Parse Size generated with respect to a reference with varying numbers of Chromosome-19 copies. Note the change at point 64 from logarithmic to linear decrease.

10. Number of Genomes to Include in Reference

Our result shows that even if returns diminish after a certain amount, there is still value should we be able to index values greater than those close to 64. However, we should note that even if we eliminate reference bias, for example reference alleles, we also increase ambiguity (Pritt, 2018). This can confuse the aligner, and drastically increase the places where a read can fall. Indeed, studies using variation graphs which prioritized the correction of reference allele bias and penalized genomes which brought the fewest unique variance found that accuracy peaked when including 8-10% of available variation (Pritt, 2018). Further, using all the variation was no better than using a single (linear) reference in terms of solving allele bias.

This data is specific to variation graphs; however, using the R-Index we reference the sequences themselves. Thus, we avoid the problem of ambiguity, in a way, that variation graphs approach since they consider all possible combinations of base alleles. The problem with early approaches, such as the FM-Index, is simply the inability to handle such high numbers of genomes in the reference to start with. Recalling our results were based on the R-Index ad it's capabilities, including at least 64 genomes in the reference maximizes the

returns of information. This seems to suggest that variation is more valuable at these points, and that at least 64 genomes does not yet diminish the return of variation with ambiguity. Of course, we can handle indexing more genomes, but this threshold, by our results, is an informed baseline. Further gain should be balanced on ambiguity and computation resources.

11. Outlook

In survey of the reference genome and genome recovery, we saw the need to use a reference to align newly sequences genomes, as De Novo assembly is too difficult a task to carry out with any regularity. However, the issues with using only a single reference genome, namely the lack of diversity and bias factors, necessitate including multiple genomes. The most common approach, the FM-Index, is not ideal for this job.

Approaches with multiple indexing must make use of the similarity between genomes in the reference, or other approaches, which make the result significantly compressible. We saw a few technologies which can avoid the ballooning memory requirements whilst maintaining speed: hybrid indexing, founder sequences, variation graphs, and the r-index. With this new technology, we are now tasked with analyzing the information gained using multiple genomes, and when it is worth it.

Using the R-Index, which can handle indexing of high amounts of genomes efficiently, we used PHONI to generate the matching statistics of a pattern of chromosome-19 with references consisting of various numbers of chromosome-19 copies. With the matching statistics, the RLZ Parse sizes could be computed for each value, evaluating the compressibility and, therefore, the information gained by using multiple copies as it relates to the introduction of helpful variation increasing similarity. These sizes descended logarithmically for values less than 64. Thus, we raise this value as a basepoint to maximize information gained when including multiple reference points.

Our result also showed a still promising linear descent in parse size, and by consequence, linear increase in compressibility and value. Although concerns exist with ambiguity, and further using other approaches, the ability of R-Index to index large numbers of reference

genomes, motivates further study into which how many sequences should be considered. Further work should help to relate the RLZ parse size result with more in-depth analysis of information gain as a whole and the result of using high numbers of genomes in the reference.

References

Ballouz, S., Dobin, A. & Gillis, J.A. Is it time to change the reference genome? *Genome Biol* **20,** 159 (2019). Retrieved December 15, 2020, from https://doi.org/10.1186/s13059-019-1774-4

Boucher, C., Gagie, T., … Rossi, M. (2020). *PHONI: Streamed Matching Statistics with Multi-Genome References.* Retrieved December 15, 2020, from https://www.researchgate.net/publication/345758535_PHONI_Streamed_Matching_Statistics_with_Multi-Genome_References

Cox, A. J., Farruggia, A., Gagie, T., Puglisi, S. J., & Sirén, J. (2016). RLZAP: Relative Lempel-Ziv with Adaptive Pointers. *String Processing and Information Retrieval Lecture Notes in Computer Science,* 1-14. doi:10.1007/978-3-319-46049-9_1

Ferrada, H., Gagie, T., Hirvola, T., & Puglisi, S. (2014, May 28). Hybrid indexes for repetitive datasets. Retrieved December 12, 2020, from https://royalsocietypublishing.org/doi/full/10.1098/rsta.2013.0137

Gagie, T. Dalhousie University. ResearchGate. (2020a) *MONI: Matching Statistics with Multi-Genome References.* Retrieved December 15, 2020, from https://www.dropbox.com/s/06mbaj4w0l16wv1/Travis_Bicocca_lecture_1.mp4?dl=0

Gagie, T., Navarro, G., & Prezza, N. (2020b). Fully Functional Suffix Trees and Optimal Text Searching in BWT-Runs Bounded Space. *Journal of the ACM, 67*(1), 1-54. doi:10.1145/3375890

K. Schneeberger, J., L. Huang, V., J. Sirén, N., A. Dilthey, C., Garrison. Erik, S., D. Valenzuela, T., . . . Simon. Gog, T. (1970, January 01). Linear time minimum segmentation enables scalable founder reconstruction. Retrieved December 12, 2020, from https://almob.biomedcentral.com/articles/10.1186/s13015-019-0147-6

Kuruppu, S., Puglisi, S. J., & Zobel, J. (2010). Relative Lempel-Ziv Compression of Genomes for Large-Scale Storage and Retrieval. *String Processing and Information Retrieval Lecture Notes in Computer Science,* 201-206. doi:10.1007/978-3-642-16321-0_20

Langmead, B. (n.d.). Langmead Lab. *Algorithms for DNA Sequencing*. Retrieved December 15, 2020, from http://www.langmead-lab.org/teaching-materia

Langmead, B. (2020). *Fighting reference bias with pangenomes*. Video. CeBIB Workship (November 23, 2020), John Hopkins University.

Pritt, J., Chen, N., & Langmead, B. (2018). FORGe: Prioritizing variants for graph genomes. doi:10.1101/311720

Sirén, J., Garrison, E., Novak, A., Paten, B., & Durbin, R. (2019, July 26). Haplotype-aware graph indexes. Retrieved December 12, 2020, from https://academic.oup.com/bioinformatics/article/36/2/400/5538990

U.S. National Library of Medicine. *Genome Reference Consortium*. National Center for Biotechnology Information. Retrieved December 15, 2020, from https://www.ncbi.nlm.nih.gov/grc.