Comparing Matching Statistics Computed with Respect to Datasets and to their Founder Sequences

by

Yansong Li

Submitted in partial fulfilment of the requirements
for the degree of Bachelor of Computer Science Honours

at

Dalhousie University
Halifax, Nova Scotia
April 2021

Dedications

To my beloved grandfather who always spoiled me the most. To my family, mentors, friends and everyone else who helped me through all this.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

It has recently been shown that the matching statistics between a DNA sample and a pan-genomic dataset can be used to estimate their similarity and that larger datasets often give more accurate estimates. In this thesis, we investigate using the dataset's so-called founder sequences instead of the datasets themselves. This is a tempting alternative because the dataset's founder sequences are much smaller and easier to handle, but still reflect the dataset's genetic diversity. Unfortunately, we found that the matching statistics obtained with the founder sequences are noticeably different from those obtained with the datasets themselves.

16 founders were generated based on 1000 copies of chromosome 19. Then, two sets of matching statistics were computed, the first set used 99 copies of chromosome 19 with respect to the 1000 copies; the second set used 99 copies of chromosome 19 with respect to the founder sequences. Matching statistics results showed the difference between the two sets of length distributions is rather significant. Over 86% of the lengths in the set of founder sequences are shorter than 10,000 which the set of 1000 chromosome 19s only has less than 23%. The overall length distribution of the founder sequences is different from the set of 1000 chromosome 19s; hence, the founder sequences cannot represent the 1000 chromosome 19s. Therefore, founder sequences are not a reliable substitute for complete datasets when estimating similarities using matching statistics.

LIST OF ABBREVIATIONS USED

BCF	Binary Counterpart Format
BWT	Burrows-Wheeler Transform
GB	Gigabyte
HG	Human Genome
KL	Kullback-Leibler divergence
LCE	Longest Common Extension
MEMs	Maximal Exact Matches
MEMs	Maximum Exact Matches
MS	Matching Statistics
NP	Nondeterministic Polynomial time
pBWT	Positional Burrows-Wheeler Transform
RAM	Random Access Memory
RMQ	Range Maximum Query
SNPs	Single Nucleotide Polymorphisms
VCF	Variant Call Format

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincerest appreciation to my supervisor Dr. Travis Gagie, for his invaluable and continuous guidance, support, and inspiration from the beginning of the directed studies to the completion of my thesis. I would also like to extend my deepest gratitude to Dr. Dominik Koeppl, Dr. Massimiliano Rossi, and Tuukka Norri, for their help during the experiment both for data generation and software usage.

Sincerely thanks to Dr. Benjamin Langmead for providing me with very valuable lectures resource during the Directed Studies. Also, many thanks to Dr. Jarno Alanko for the guidance and advice for the directed studies project as well.

Last but not least, the completion of my bachelor's degree would not have been possible without the unconditional support and unparalleled love from my parents and grandparents.

SECTION 1 INTRODUCTION

Genome sequencing is a major research field in bioinformatics that aims for discovering the specific DNA sequence compositions of organisms' genomes of different species. Computing the matching statistics between a sample of DNA sequences and a pan-genomic dataset has gradually become a valuable approach in the field of genome sequencing, which helps the bioinformatics specialists estimate the similarity of them by locating the occurrence of the longest prefixes of the set of DNA sequences in the pan-genomic dataset, and it also records the length of the matched longest prefixes (Boucher, 2021).

More specifically, the computation of matching statistics has two parameters, the pattern and text, which both are arrays of characters (Boucher et al., 2021). Meanwhile, the matching statistics is a vector of pairs, each pair is formed as (position, length) (Boucher et al., 2021). According to the PHONI paper from Boucher et al., the matching statistics are defined as the following.

1. $P[i \ldots i + MS[i].len - 1] = T\,[MS[i].pos \ldots MS[i].pos + MS[i].len - 1]$
2. $P[i \ldots i + MS[i].len]$ *does not occur in T.*

where $MS[i].pos$ is a pointer in the text T, which points to the beginning position of the "longest prefix of $P[i \ldots x - 1]$ that occurs in T"; $MS[i].len$ is the length of that prefix (Boucher, 2021).

Moreover, computing the matching statistics of multiple genomes with respect to a large database requires a huge amount of RAM and time, which is not realizable in most scenarios. Lately, Boucher et al. have designed and implemented software for computing the streamed matching statistics of long patterns, which reduces the amount of RAM that is used for matching statistics computation significantly (2021). With the support of such software, using a number of genomes as the input text is no longer the biggest concern in matching statistic computations.

The distribution of lengths of the matched longest prefixes in matching statistics helps the bioinformatics specialists to determine the correlation of the input text and input patterns. Nevertheless, a matching statistics computation based on a number of genomes with respect to a complete genome database would require a large amount of RAM and running time. For example, a database that contains 2504 copies of chromosome 19 takes 56 GB of space if it is stored in a FASTA file (IGSR, 2021) with algorithms. While the bioinformatics specialists are trying to design more effective algorithms and data structures for matching statistics computations, the cost of computation could be reduced from the original inputs. Instead of using an entire database of DNA sequences as the input text, the bioinformatics specialists could use the founder sequences of the original database.

Ideally, the founder sequences are the DNA sequences that “belong to a small number of settlers who founded the population some generations ago” (Ukkonen, 2002). It is presumed that “the population was evolved independently, but with a controllable number of point mutations” (Ukkonen, 2002). Moreover, the current DNA sequences in the database are the recombination of the “corresponding sequences of the founders” (Ukkonen, 2002). Therefore, the bioinformatics specialists are able to find a set of founder sequences with “plausible reconstructions” of the DNA sequences in the database (Ukkonen, 2002).

Once the set of founder sequences has been constructed from the database of DNA sequences, the bioinformatics specialists can replace the database originally used as the text with the founder sequences for the matching statistics computation if the length distribution of the founder sequences is not dramatically different from the length distribution of the original DNA sequences, which means two sets of lengths can demonstrate similar trends on histograms. Then the cost of computing the matching statistics will decrease because the founder sequences are much smaller. Therefore, indexing large collections of genomes will not be necessary for matching statistics anymore.

In order to testify whether a set of founder sequences can replace the database as the text in matching statistics computation to some extent, the following experiment has been conducted as the first stage of the research.

1. Extract 1000 copies of chromosome 19 from the database of the International Genome Sample Resource in VCF format as a pan-genomic dataset(IGSR, 2021).
2. Build a set of founder sequences of the 1000 copies of chromosome 19.
3. Extract another 99 copies of chromosome 19 from the same database in VCF format.
4. Compute the matching statistics of the 99 copies of chromosome 19 with respect to the 1000 copies of chromosome 19 using PHONI.
5. Compute the matching statistics of the 99 copies of chromosome 19 with respect to the set of founder sequences using PHONI.
6. Compare the length distributions of the two sets of matching statistics.

Fewer long matches in the set of founder sequences are expected, but the founder sequences will still be a satisfying replacement if the difference between the two sets of length distributions is not too significant. Otherwise, the necessity of computing matching statistics using large numbers of genomes as the input text has been testified.

For the rest of this paper, articles related to the topic of matching statistics computation and founder sequences generation will be further analyzed and summarized in section 2. Next, a more detailed and specific description of the experiment will be illustrated such as the software and methods that are involved in section 3. After that, the result of the experiment will be stated and discussed in section 4. Finally, a conclusion will be made based on the results of the experiment and the direction of future research will be indicated.

SECTION 2 LITERATURE REVIEW

2.1 Matching statistics and cross-entropy.

The matching statistics of the string *P[0...x – 1]* with respect to the text *T[0…y - 1]* is an array of pairs *MS[0...x – 1]* that "stores at position *i* the length of the longest prefix of *P[i…x – 1]* that occurs at least once in *T*" (Mäkinen, 2015). The *MS* array with respect to input pattern *P* and input text *T* can be used to define the text-similarity between pattern *P* and text *T* without "resorting to alignment" and without "specifying a fixed string length *k*" (Mäkinen, V, 2015). Meanwhile, this *MS* array of pairs helps the bioinformatics specialists to evaluate the cross-entropy of processes that randomly generated the pattern *P* and text *T* (Mäkinen, 2015).

Cross-entropy has an inseparable relationship with Shannon entropy, which is "defined for a given discrete probability distribution" (Green, 2016). The average number of bits per character required for random sample identification with finite memory from a given discrete probability distribution is the Shannon entropy (Green, 2016). Besides this, there is no coding scheme that is completely defined by Shannon entropy, but it does define "how much information an optimal coding scheme for a given distribution would use, on average, to identify random samples from that distribution." (Green, 2016). For instance, there are two distributions *D* and *D'* that share the same set of symbols, but the "probabilities that are assigned on the symbols of the two distributions are different" (Green, 2016). Similar to Shannon entropy, the cross-entropy is also defined for the discrete probabilities distributions *D* and *D'* (Using the same character for two distributions shows that they share the same alphabet) (Green, 2016). Moreover, it also estimates the average number of bits per character required to "identify the random sample from *D'* when using an optimal coding scheme constructed for *D*." (Green, 2016). The mathematical cross-entropy equation that represents the distribution *D* and *D'* is the following (Green, 2016).

$$H(D', D) = -\sum_i P(D'_i) \log P(D_i)$$

The log notation in the cross-entropy equation expresses the "optimized code length of a particular probability", which means the distribution *D* is "substituted in (i.e. the optimal coding scheme for D is in use)" (Green, 2016). Meanwhile, the probability of each individual symbol is represented by the left-hand probability expression, so the other distribution *D'* is also "substituted in" (Green, 2016). Also, the "Kullback-Leibler divergence" is another related notion of two probability distributions, which is represented by the following (Mäkinen, 2015).

$$KL(D', D) = \lim_{i \to \infty} \frac{-\sum_i P(D'_i)(\log P(D'_i) - \log P(D_i))}{n}$$

$$KL(D', D) = H(D', D) - H(D')$$

More specifically, it determines the "expected number of extra bits per character" needed for a string identification generated by distribution *D* using a coding scheme derived from distribution *D'* (Mäkinen, 2015). Furthermore, d_{KL}*(D', D) = KL(D', D) + KL(D, D')*, which can be used to estimate the dissimilarity d_{KL}*(D', D)* between two probability distributions, it is " symmetric, non-negative and equals to zero when *D'* is the same as *D*" (Mäkinen, 2015).

Assume a finite sequence *T* has *m* observations from T_1 to T_m of the "random process with probability distribution" *D* is applied, and another sequence *P* with n observations from P_1 to P_n is receiving from the "random process with probability distribution" *D'* (Mäkinen, 2015). The length of the longest prefix of *P* that occurs at least once in *T* is represented by *l*, the expected value of *l* with respect to the distribution D and D' is symbolized as $E_{D\prime,D}(l)$, and the mathematical equation is shown as the following (Mäkinen, 2015).

$$\lim_{m\to\infty}\left|E_{D\prime,D}(l)-\frac{\log m}{H(D',D)}\right|\in O(1)$$

Hence, the cross-entropy of distribution *D* and *D' H(D', D)* can be evaluated by $\log m\,/E_{D',D}(l)$. For a sequence "is finite but long enough" such P, the length of such a longest prefix *l* can be evaluated by,

$$\tilde{l}=\sum_{i=1}^{m}MS_{D\prime,D}[i]/m$$

where $MS_{D\prime,D}$ represents the vector of the matching statistics of *P* with respect to *T* (Mäkinen, 2015). Such estimators are known as "average common substring estimators" for H(D' D), KL(D', D), and d_{KL}*(D', D)* (Mäkinen, 2015).

2.2 The Burrows-Wheeler Transform.

Burrows-Wheeler Transform is a tool for data comparison, a string of characters is transformed to its BWT by listing all the permutations of it in alphabetical order, which is formed Burrows-Wheeler Matrix; then the last column of the transformation is the BWT (Langmead, 2020b). More specifically, each character in the first column shares the same rank with each character in the last column (BWT) (Langmead, 2020b). Such a property is called the LF-Mapping, where the rank is the numbered occurrence of that specific character appears (Langmead, 2020b). For instance, if a third A occurs in the first column denoted as A_3, which can be found by locating the third A in the last column. This property follows from the lexicographical sorting alongside the definition of right-context (Langmead, 2020b).

With the LF-Mapping, only the first and last (BWT) column of the Burrows-Wheeler Matrix need to be stored since the content of the original string can be recovered by traversing these two columns then reversing the that string(Langmead, 2020b). More specifically, the procedure starting at the first element of the first column $\$$(terminating character); then locating the preceding character in the same row but in the last column of the matrix (Langmead, 2020b). After that, jump to the first column and find the current preceding character with the same rank. Next go to the last column find the next preceding character(Langmead, 2020b). By repeating such a process until encounter next $\$$ using only the two columns. Finally, reverse the current string to get the original string. (Langmead, 2020b).

2.3 The FM-index based on BWT.

The FM-index is a strategy that answers full-text indexing queries of a large amount of text, which was developed based on BWT algorithm and other auxiliary structures (Langmead, 2020b). More specifically, such a full-text index of “text $T \in \sum^{n}$ is a structure giving efficient answers to queries” (Langmead, 2020b).

1. “$Locate(P)$, returns all offsets where $P \in \sum^{m}$ matches a substring of T.
2. $Count(P)$ returns # of offsets where *P* matches a substring of *T*.
3. $Extract(i, m)$ returns $T[i{:}\, i + m - 1]$ (length-m substring starting at *i*)” (Langmead, 2020b)

Armed with efficient rank data structure, the FM-Index is highly compressible, sped up by wavelet tree data structure (Langmead, 2020b). 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 return the number of these offsets (Langmead, 2020b). The core of performing these queries is starting with the shortest suffix of the pattern and matching “successively longer patterns” (Langmead, 2020b). For example, given the initial pattern *A*, then finding the rows starting with *A* using LF-Mapping to find the next shorter pattern; if the suffix is *BA*, find where a *B* preceded this *A* until the whole pattern is matched (Langmead, 2020b). To return the indices by locating queries, a sampled suffix array holding the actual indices is stored because storing the whole array would be costly, leveraging that if the match has no value in the suffix array, using the LF-Mapping to find the next sampled index (Langmead, 2020b).

2.4 The pan-genome.

Since a single reference genome is not particularly representative of the full genetic variation of a species because it can cause reference bias, cataloguing as much human genetic variation as possible can ensure the results are more accurate than single reference genome. Hence, references that use multiple genomes are of benefit, which is also referring as pan-genome. More specifically, pan-genome 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, they can also study ancient humans’ genomes, tumors, bacteria, and how genomes work. The pan-genome plays an essential guide role in all these problems (Langmead, 2016).

Recently, the National Human Genome Research Institute has initiated research to build a reference genome based on 350 different individual organisms (Sherman, 2020). However, there is no existing "comprehensive, analyzable human pan-genome that surveys a wide variety of human populations, captures both genic and intergenic variation, and incorporates this variation into a single utilizable pan-genome" of human yet, even with increasingly more on-going "large-scale human sequencing projects" (Sherman, 2020). In addition, the creation of "population-specific pan-genomes" is also a cutting-edge topic in bioinformatics because of the variation of "human SNPs and structural variants" (Sherman, 2020). Eventually, a reference genome that can represent the entire human species for "pan-genomic analyses" will emerge along with the development of "computational methods capable of handling larger and larger datasets" (Sherman, 2020).

2.5 The PHONI's predecessors and its improvements.

Finding an effective approach of indexing a "compressed representation of a massive and highly repetitive text" in the field of matching statistics computation was a problem that Boucher et al. have been working on for years (Boucher, 2021). For example, computing the matching statistics of "a given pattern concerning a database of genomes of individual organisms of the same species" (Boucher, 2021). Gagie et al.'s r-index accelerated the computation of matching statistics. Based on Gagie et al.'s implementation, MONI was implemented for aligning reads to a multi-genome reference as its final but unaccomplished goal. The software was built by Rossi et al., and its primary function is finding " MEMs between a set of DNA reads ad a genomic database" (Boucher, 2021). For the ultimate goal of MONI, Bannai et al. have designed a solution that makes two passes over the pattern. Specifically, the first traverse of the pattern is from the right to left, which "computes and buffers the position *pos* values of the MS array" (Boucher, 2021). Next, the second traverse is from the left to right, which "uses the *pos* value from last pass and random access to the text *T* to compute the length *len* values of the MS array" (Boucher, 2021).

Nevertheless, such an algorithm has a serious flaw when the patterns are being given online, character by character because the latency for the computation "grows linearly with the length of the pattern" (Boucher, 2021). Also, the match statistics of complete genomes concerning genomic databases is critical because such MS data of whole genomes could be beneficial for "genetic diversity evaluation or pathogens mutation in a population" (Boucher, 2021). However, the maximum latency, "from the time that a character is received to the time the corresponding pair has been returned" of Bannai et al.'s solution is proportional to the growth of the length of the pattern, which means the required time for generating the corresponding pairs of the matching statistics grows along with the time that receiving the patterns character by character online (Boucher, 2021). Hence, the current MONI software is not capable of receiving whole genomes as the patterns for matching statistics computation online.

Anyway, the MONI will be a powerful tool for computing matching statistics of DNA sequencing or other long patterns if it can stream the result online. In detail, such software will be able to stop the output in time if it becomes "incompressible relative to the database" or unrelated to the current purposes because it can rapidly return the computation feedback. (Boucher, 2021). Meanwhile, the users also prefer "a rapid stop when the software has found a long enough match to confirm the presence of a pathogen”. (Boucher, 2021). Therefore, on top of the current functions of MONI, features such as reducing latency, optimizing throughput, and reducing memory footprint can be expected to deploy matching statistics computation on machines (Boucher, 2021).

With the intensions described above, a new tool PHONI was implemented based on MONI by Boucher et al., which process the patterns only with “one single traverse and make it streaming”. (Boucher, 2021). The software uses LCE queries for the length *len* values generation while computing the position *pos* values in the first traverse of matching statistics computation (Boucher, 2021). Compared to the two passes algorithm of MONI, Boucher et al.’s implementation requires considerably less memory and time to build and uses significantly less extra RAM for matching statistics computation of extremely long patterns (Boucher, 2021).

2.6 The reconstruction of founder sequences.

One of the ongoing researches in the field of pan-genomics is to “develop a sufficiently small, efficiently queryable, but still a descriptive representation of the entire genome database” (Norri, 2019). Therefor, the founder sequences reconstruction problem has been purposed when it comes to DNA sequences. The primary feature of the founder sequence is representing an enormous size of DNA sequences with a small set of sequences and preserve the contiguities of the original DNA sequences as much as possible (Norri, 2019).

More precisely, if there are original *m* DNA sequences in the database, then a set of *d* sequences that are reconstructed from it with the lowest number of crossovers is the founders of the original *m* sequences, which *d* is an appropriate number that was chose by the software. For example, given a set of *m* DNA sequences $C = \{C_1, C_2, \ldots, C_m\} \subseteq \sum^n$ with length n over alphabet $\sum = \{A, C, G, T\}$. Then a set of k sequences $F = \{F_1, F_2, \ldots, F_k\} \subseteq (\sum \cup \{-\})^n$ would be the founder set of C (Ukkonen, 2002). The $"-"$ symbol points out the unreconstructed locations in the founder sequences, but it can not occur in a recombinant sequence of C (Ukkonen, 2002). However, such an assumption is based on each C_i in the original set C has a parse in terms of all founder sequences, which means each C_i in C has been decomposed into a number of "non-empty snippets f_{ih} such that $C_i = f_{i1} f_{i2} \ldots f_{ip_i}$" and each snippet can be found in "some $F_j \in F$ with exactly the same position as in C_i" where snippets mean the short reads from C (Ukkonen, 2002).

"Optimizing a set of founders of DNA sequences is an NP-hard problem even to approximate within a constant factor" (Norri, 2019), and the primary difficulty of the founder reconstruction is the problem of *minimum set of founders* . Nevertheless, there is still a polynomial-time of solution for this problem using segmentation because the "minimum founder set problem and the minimum segmentation problem are connected" (Ukkonen, 2002). The solution was first purposed by Ukkonen using a dynamic programming algorithm with the time complexity of $O(n^2 m)$ in 2002. More explicitly, there is a size limitation *M* for the founder sequences (Ukkonen, 2002). Then, the founder sequences *F* that have the "maximum average fragment length λ_{ave} with its corresponding parse" can be constructed as the following (Ukkonen, 2002). The set of all consistent *M*-partitions is denoted as P_j for the rows of *C* at column *j* (Ukkonen, 2002). Such partitions could be divided into *M* classes, which means they are sub-partitions of the partition “induced by the equality of the symbols on column *j*”; also, they are possibly empty (Ukkonen, 2002). Then two *M*-partitions *P and P’* of continuous columns are collected using the minimum amount *U(P, P’)* of crossovers between the columns (Ukkonen, 2002).

For the smallest possible number of crossovers in *M*-colorings of *C[1, j]* that have partition $P \in P_j$ at column j, it is represented by *S(j, P)*. The mathematical expression is shown as the following (Ukkonen, 2002).

$$S(j,P) = \min_{P' \in P_{j-1}} \{S(j-1,P') + U(P',P)\}$$

Hence, the small number of crossovers is $S(n) = \min_{P \in P_n} S(n,P)$, and the corresponding parse has the maximum average fragment length $\lambda_{ave} = mn/(S(n,P)+m)$ (Ukkonen, 2002).

Then the minimum crossovers *S(j, P)* can be evaluated using an algorithm that its processing time grows linear with *nN* where “*N* is the length of the largest *M*-partitions P_j” (Ukkonen, 2002). The bound N could be exceeded when the number of the DNA sequences *m* and the bound M of the size of founder sequences grows too much, but such a special case would only occur when the parameter is too big (Ukkonen, 2002).

As a result, the problem of minimizing the set of founder sequences can be solved using the minimum crossovers equation with the condition of the maximum average fragment length λ_{ave} is larger than the fragment length bound L. The minimum bound M with the corresponding λ_{ave} is at least L if a binary search is applied for the set of DNA sequences from 1 to *m* (Ukkonen, 2002). Ukkonen’s approach works functionally if the λ_{ave} increases monotonically with the bound *M*, but the time complexity is still the square of the size of the alphabet *n* multiples the number of the original DNA sequences *m*

$O(mn^2)$, which still can be improved. Therefore, Norri et al. have designed a new algorithm with the time complexity $O(mn)$ for the minimum segmentation problem and implemented software for the founder sequence generation that uses the algorithm of Norri et al. in 2019.

2.7 The improved reconstruction of founder sequences.

The improvement of the new algorithm is primarily based on the positional BWT with larger alphabets (Norri, 2019). The dynamic programming solution that was purposed by Dr. Ukkonen requires $O(mn^2)$ time complexity for the entire computation and $O(mn)$ for searching the "best preceding segment boundary for each column of the input" (Norri et al., 2019). Nevertheless, Norri revealed that the minimum crossovers can be stored directly in pBWT internal structure with some minor modifications (Norri, 2019). Precisely, "only the latest L values computed by the dynamic programming are required, which means the updated time complexity is $O(m + L)$", where L is the "input threshold on the length of each segment" (Norri, 2019). Next, the segmentation can be reassembled using "the standard backtracking approach in the time complexity of $O(n)$ with a vector of length n" (Norri, 2019).

According to Norri et al. the positional Burrows-Wheeler transform is a pair of integer arrays $a_k[1, m]$ and $d_k[1, m]$ of a set of recombinants $C = \{C_1, C_2, \ldots, C_m\} \subseteq \sum^n$ with the following two conditions.

1. " $a_k[1, m]$ is a permutation of $[1, m]$ such that $C'_{a_k[1],k} \leq \cdots \leq C'_{a_k[n],k}$ lexicographically, where $C'_{i,k}$ is the reversal of $C_i[1, k]$ and $k \in [1, n]$"
2. "For $k \in [1, n]$, $d_k[i]$ is an integer such that $C_{a_k[i]}[d_k[i], k]$ is the longest common

suffix of $C_{a_k[i]}[1,k]$ and $C_{a_k[i-1]}[1,k]$, and $d_k[i]$ = k + 1 if either this suffix is empty or *i* = 1." (Norri, 2019)

The advantage of the pBWT is that it uses "at most m values to determine segment boundaries where the number of distinct founder strings change while the original solution uses *O(nm)* time to look for the best preceding segment boundary for each column of the input" (Norri, 2019).

In order to decrease the time complexity of computing the minimum segmentation, the generation of a_k and d_k from a_{k-1} and d_{k-1} in the pBWT was revised to use range maximum query (RMQ) by an algorithm that was developed by Norri. Such an algorithm is capable of reducing the time complexity of generating pBWT to $O(m \log |\sum|)$ where $[0, |\sum| - 1]$ is the alphabet with length $O(m)$ (Norri, 2019).

There are *n* distinct characters in the alphabet of the original DNA sequences set; therefore, the overall time complexity of computing the smallest number of founders of a set of DNA sequences would be $O(mn^2)$ (Norri, 2019). After that, the backtracking approach will be applied for identifying the optimal segmentation (Norri, 2019). Then the samples will be updated to the "locations of segment boundaries" since they were stored for the building of the pBWT arrays, which were used for the distinct substrings' identification in each segment (Norri, 2019). Next, the process of "joining the adjacent segments" will be initiated from left to right with the condition that "the number of distinct substrings in a single segment is smaller than the minimum number of founders" (Norri, 2019). Finally, the founder sequences can be generated by concatenating the substrings (Norri, 2019). Overall Norri et al.'s implementation reduces the time complexity of the minimum segmentation of founder reconstruction from $O(mn^2)$ to $O(mn)$, which effectively expands the input size of the founder sequences reconstruction to thousands of complete human chromosomes.

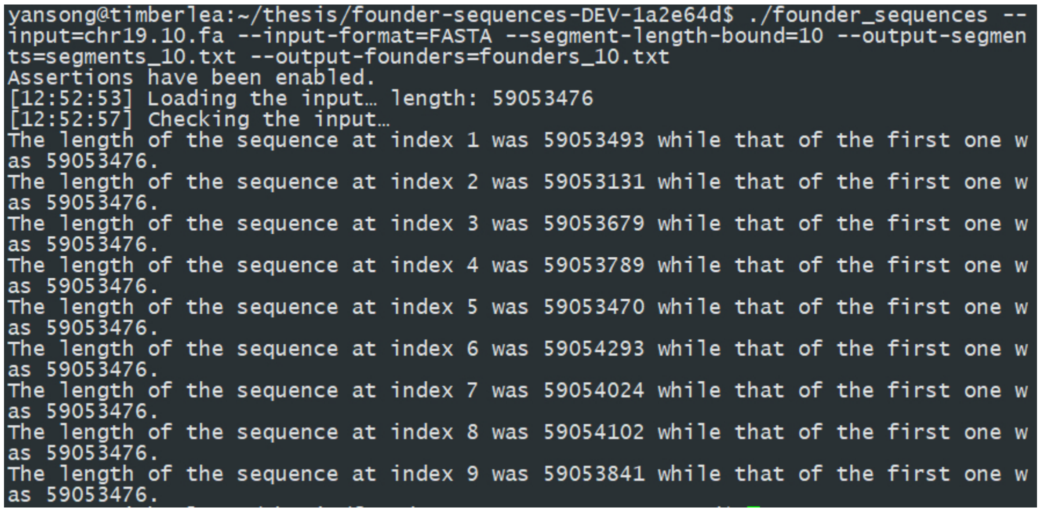
SECTION 3 MATERIALS AND METHODS

3.1 Founder sequences generation with FASTA file.

Following the suggestion from Norri, the experiment was initiated with his software "founder-sequences", which "creates a set of founder sequences from a set of multiple-aligned sequences", and the acceptable input file format of "founder-sequences" are text and FASTA (Norri, 2019). 10 copies and 1000 copies of chromosome 19 were collected from 1000 Genome Project by Dr. Dr. Massimiliano Rossi (IGSR, 2019). For simplicity, the following command was attempted for the generation of founder sequences of 10 copies of chromosome 19:

./founder_sequences --input=chr19.10.fa --input-format=FASTA --segment-length-bound=10 --output-segments=segments_10.txt --output-founder=founders_10.txt

However, the software indicated that the DNA sequences in the FASTA file are not aligned as shown in Figure3.1, which is not acceptable because aligned sequence is mandatory as the input of the software.



```
yansong@timberlea:~/thesis/founder-sequences-DEV-1a2e64d$ ./founder_sequences --
input=chr19.10.fa --input-format=FASTA --segment-length-bound=10 --output-segmen
ts=segments_10.txt --output-founders=founders_10.txt
Assertions have been enabled.
[12:52:53] Loading the input… length: 59053476
[12:52:57] Checking the input…
The length of the sequence at index 1 was 59053493 while that of the first one w
as 59053476.
The length of the sequence at index 2 was 59053131 while that of the first one w
as 59053476.
The length of the sequence at index 3 was 59053679 while that of the first one w
as 59053476.
The length of the sequence at index 4 was 59053789 while that of the first one w
as 59053476.
The length of the sequence at index 5 was 59053470 while that of the first one w
as 59053476.
The length of the sequence at index 6 was 59054293 while that of the first one w
as 59053476.
The length of the sequence at index 7 was 59054024 while that of the first one w
as 59053476.
The length of the sequence at index 8 was 59054102 while that of the first one w
as 59053476.
The length of the sequence at index 9 was 59053841 while that of the first one w
as 59053476.
```

Figure 3.1: Generating founder sequences of 10 copies of chromosome 19 with FASTA file.

Therefore, a multi-sequence aligner was needed to preprocess the input FASTA file. The Clustal Omega was applied for the alignment, which is a multi-sequence aligner that is currently maintained at the Conway Institute UCD Dublin (Higgins, 2021); however, it turned out the running time of Clustal Omega only grows linearly concerning the number of sequences in the FASTA file, and it took more than hours to just align 10 copies of chromosome 19. Hence, the multi alignment of 1000 copies was infeasible using Clustal Omega, and a more effective multi sequences aligner must be applied. Unfortunately, there was no more available powerful multi sequences aligners that can be accessed, and Tuukka Norri suggested another founder sequences generator that he developed, which use VCF files as input.

3.2 Extracting 1000 copies of chromosome 19 from the IGSR database.

The DNA sequences of human genomes are available on the website of 1000 Genome Project in various formats (IGSR, 2021). The VCF file of chromosome 19 was collected using the following command:

wget -O ALL.chr19.shapeit2_integrated_snvindels_v2a_27022019.GRCh38.phased.vcf.gz ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/data_collections/1000_genomes_project/release/20190312_biallelic_SNV_and_INDEL/ALL.chr19.shapeit2_integrated_snvindels_v2a_27022019.GRCh38.phased.vcf.gz

Also, the file's name has been modified to ALL.chr19.indels.vcf.gz for convenience of future use.

The Bcftools was introduced for multiallelic variants elimination and copies extraction, which has “a set of utilities that manipulate variant calls in the Variant Call Format (VCF) and its binary counterpart” (Li, 2020).

For removing the multiallelic variants in the original VCF file, the following command was applied using bcftools:

```
bcftools view -v snps,indels -m2 -M2 -Oz -o  ALL.chr19.indels.vcf.gz
```

The phase 3 release of the 1000 Genome Project from IGSR has 2504 individuals, which means extracting 1000 copies of chromosome 19 would be needed as the preparation of matching statistics computations (IGSR, 2021). Otherwise, there will be no copies left as the input patterns in the computations.

The process of copies extraction started with storing all the names of 1000 targeted samples. The following command was applied to store all the samples’ names:

```
./bcftools query -l ALL.chr19.indels.vcf > names.txt
```

2504 sample names were stored into the names.txt file, and 1000 of them were manually stored into names_1000.txt using the Emacs from HG00096 to HG02549 (There are gaps between sequences). Also, another 99 copies’ names were stored into names_99.txt from HG02554 to HG02782 for the input patterns of matching statistics computations.

With the targeted samples’ names stored in *.txt files, those corresponding 1000 copies and 99 copies can be extracted from the original VCF file of the IGSR database. The 1000 copies of chromosome 19 were extracted using the following command and stored in a VCF file:

```
./bcftools view -c1 -S names_1000.txt -Oz --threads 8 -o chr19.1000.indels.vcf.gz
ALL.chr19.indels.vcf.gz
```

Similarly, the 99 copies of chromosome 19 was extracted using the following command and stored into a VCF file:

```
./bcftools view -c1 -S names_99.txt -Oz --threads 8 -o chr19.99.indels.vcf.gz
ALL.chr19.indels.vcf.gz
```

For double-checking the number of copies in the VCF files, the following commands were applied with respect to 1000 copies and 99 copies using Bcftools:

```
./bcftools query -l chr19.1000.indels.vcf.gz | wc -l
./bcftools query -l chr19.99.indels.vcf.gz | wc -l
```

Both commands indicated the numbers of copies are 1000 and 99 as expected.

So far, there are two VCF files prepared for the matching statistics computation. Specifically, the VCF file that contains 1000 copies will work as the input text and the VCF file that contains 99 copies will work as the input pattern. Meanwhile, the 1000 copies will also work as the sequences database for the founder sequences generation.

3.3 Founder Sequences Generation.

Before computing the founder sequences concerning the 1000 copies of chromosome 19, a reference sequence FASTA file of chromosome 19 is also needed for the computation as the software README file instructed. The reference sequence file can be downloaded from the link at the top of the VCF file after “#reference” using the following command.

```
wget -O GRCh38_full_analysis_set_plus_decoy_hla.fa
ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/reference/GRCh38_reference_genome/GRCh38_full_analysis_set_plus_decoy_hla.fa
```

Then a single reference of chromosome 19 can be extracted using the following command.

```
awk '$0 ~ "^>" { match($1, /^>([^:]+)/, id); filename=id[1]} {print >> filename".fa"}'
GRCh38_full_analysis_set_plus_decoy_hla.fa
```

Next, the founder sequences generation can be initiated with the FASTA file of the reference sequence of chromosome 19 and the VCF file of 1000 copies of chromosome 19 prepared. The generation starts with preprocessing the VCF file using *preprocess_vcf*. This tool takes the reference sequence FASTA file and the VCF file that contains the sequences as its inputs. It outputs " a list of optimal cut positions in the binary format", which "minimize the number of paths in each segment of the graph" (Norri, 2019). The following command was applied for the cut position computation:

```
./preprocess_vcf --reference=chr19.fa --variants=chr19.1000.indels.vcf --
output=chr19.indels.pos --chromosome=19
```

The preprocessing took 7 minutes for running, and it successfully outputted the cut position file *chr19.indels.pos*.

The second step of the generation is creating the variant graph using the create_variant_graph tool. It takes three files as its inputs, which are the FASTA file that contains the reference sequence of chromosome 19, the VCF file that has 1000 copies of chromosome 19, and the cut position file that was generated by *preprocess_vcf* from the last step. Then it outputs "the variants as a directed acyclic graph in a binary format" (Norri, 2019). The following command was applied for this step:

./create_variant_graph --reference=chr19.fa --variants=chr19.1000.indels.vcf --output=chr19.indels.graph -P --cut-position=chr19.indels.pos

The variant graph generation took 8 minutes, the output file was named *chr19.indels.graph*.

The final step of the founder sequences generation is based on the previous variant graph file and the reference sequence FASTA file. A new tool called *variant_graph_to_sequences* was involved. It outputs a certain number of founder sequences which is able to cover all the sequences in the VCF file. Each sequence is store in an independent file within the current working directory (Norri, 2019). The following command was applied for the final founder sequence generation:

./variant_graph_to_sequences --reference=chr19.fa --variants=chr19.indels.graph -F

As a result, the tool generated 16 founder sequence files that each contains a complete founder sequence of the 1000 copies of chromosome 19. The number of the founder sequences is determined by the software automatically.

3.4 Miniconda installation for running PHONI.

Since the experiment was conducted on a server that belongs to the university, new software installation using "sudo" command was not allowed due to permission. Hence, Dr. Massimiliano Rossi suggested using Miniconda to bypass the permission problem.

Once the Miniconda has been downloaded, installed, and verified using the instructions from the website, then the creation of the PHONI environment can be initiated using the following command:

conda create --name phoni-env "cmake>=3.15" "gcc_linux-64>=7.3.0" "gxx_linux-64>=7.3.0" zlib git

The versions of *cmake*, *gcc*, and *gxx* can be various based on the circumstances of the server. Next, the PHONI environment can be activated using the following command:

conda activate phoni-env

Also, the PHONI environment "*phoni-env*" needs to be reactivated every time before building and executing the PHONI software.

For the process of building PHONI, the repository was first cloned from GitHub using:

git clone https://github.com/koeppl/phoni.git

Then it was built with the following commands:

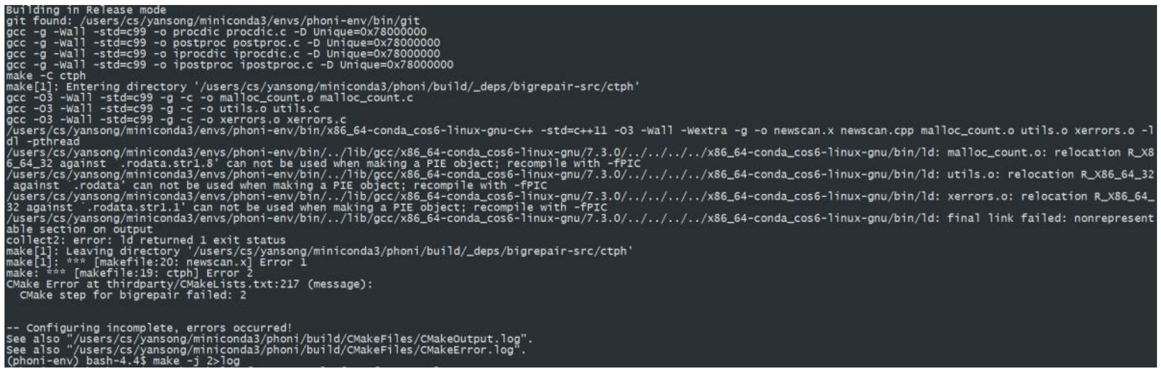
cd phoni

mkdir build

cd build

cmake -DCMAKE_BUILD_TYPE=Release ..

However, a fatal error occurred during the *cmake* process as Figure 3.2 shown, which indicated the compiled code was mixed up from two different compliers.



```
Building in Release mode
git found: /users/cs/yansong/miniconda3/envs/phoni-env/bin/git
gcc -g -Wall -std=c99 -o procdic procdic.c -D Unique=0x78000000
gcc -g -Wall -std=c99 -o postproc postproc.c -D Unique=0x78000000
gcc -g -Wall -std=c99 -o iprocdic iprocdic.c -D Unique=0x78000000
gcc -g -Wall -std=c99 -o ipostproc ipostproc.c -D Unique=0x78000000
make -C ctph
make[1]: Entering directory '/users/cs/yansong/miniconda3/phoni/build/_deps/bigrepair-src/ctph'
gcc -O3 -Wall -std=c99 -g -c -o malloc_count.o malloc_count.c
gcc -O3 -Wall -std=c99 -g -c -o utils.o utils.c
gcc -O3 -Wall -std=c99 -g -c -o xerrors.o xerrors.c
/users/cs/yansong/miniconda3/envs/phoni-env/bin/x86_64-conda_cos6-linux-gnu-c++ -std=c++11 -O3 -Wall -Wextra -g -o newscan.x newscan.cpp malloc_count.o utils.o xerrors.o -l
dl -pthread
/users/cs/yansong/miniconda3/envs/phoni-env/bin/../lib/gcc/x86_64-conda_cos6-linux-gnu/7.3.0/../../../../x86_64-conda_cos6-linux-gnu/bin/ld: malloc_count.o: relocation R_X8
6_64_32 against `.rodata.str1.8' can not be used when making a PIE object; recompile with -fPIC
/users/cs/yansong/miniconda3/envs/phoni-env/bin/../lib/gcc/x86_64-conda_cos6-linux-gnu/7.3.0/../../../../x86_64-conda_cos6-linux-gnu/bin/ld: utils.o: relocation R_X86_64_32
 against `.rodata' can not be used when making a PIE object; recompile with -fPIC
/users/cs/yansong/miniconda3/envs/phoni-env/bin/../lib/gcc/x86_64-conda_cos6-linux-gnu/7.3.0/../../../../x86_64-conda_cos6-linux-gnu/bin/ld: xerrors.o: relocation R_X86_64_
32 against `.rodata.str1.1' can not be used when making a PIE object; recompile with -fPIC
/users/cs/yansong/miniconda3/envs/phoni-env/bin/../lib/gcc/x86_64-conda_cos6-linux-gnu/7.3.0/../../../../x86_64-conda_cos6-linux-gnu/bin/ld: final link failed: nonrepresent
able section on output
collect2: error: ld returned 1 exit status
make[1]: Leaving directory '/users/cs/yansong/miniconda3/phoni/build/_deps/bigrepair-src/ctph'
make[1]: *** [makefile:20: newscan.x] Error 1
make: *** [makefile:19: ctph] Error 2
CMake Error at thirdparty/CMakeLists.txt:217 (message):
  CMake step for bigrepair failed: 2


-- Configuring incomplete, errors occurred!
See also "/users/cs/yansong/miniconda3/phoni/build/CMakeFiles/CMakeOutput.log".
See also "/users/cs/yansong/miniconda3/phoni/build/CMakeFiles/CMakeError.log".
(phoni-env) bash-4.4$ make -j 2>log
```

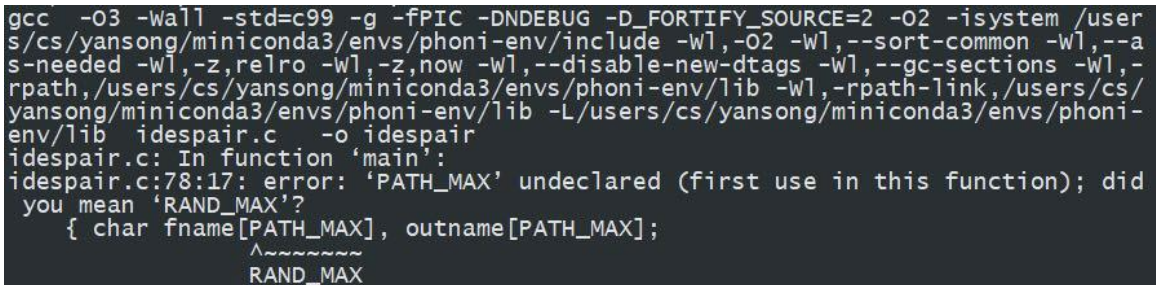
Figure 3.2: A fatal error of mixed compliers while running *cmake*.

As Dr. Koeppl suggested, the *CXX_FLAGS* and *CFLAGS* of five *makefile* that in directory *bigrepair-src* or its subdirectories were modified with *-fPIC*; also, the path to *C* compiler and *C++* compiler added to the *makefile* as the following:

CC=”/users/cs/25ansong/miniconda3/envs/phoni-env/bin/x86_64-conda_cos6-linux-gnu-cc”

CXX=”/users/cs/25ansong/miniconda3/envs/phoni-env/bin/x86_64-conda_cos6-linux-gnu-c++”

The problem of mixed compilers got resolved with the modifications of *makefile*, but a new problem occurred when re-running *cmake*. A variable *PATH_MAX* cannot be found as Figure 3.3 shown.



```
gcc  -O3 -Wall -std=c99 -g -fPIC -DNDEBUG -D_FORTIFY_SOURCE=2 -O2 -isystem /user
s/cs/yansong/miniconda3/envs/phoni-env/include -Wl,-O2 -Wl,--sort-common -Wl,--a
s-needed -Wl,-z,relro -Wl,-z,now -Wl,--disable-new-dtags -Wl,--gc-sections -Wl,-
rpath,/users/cs/yansong/miniconda3/envs/phoni-env/lib -Wl,-rpath-link,/users/cs/
yansong/miniconda3/envs/phoni-env/lib -L/users/cs/yansong/miniconda3/envs/phoni-
env/lib  idespair.c   -o idespair
idespair.c: In function ‘main’:
idespair.c:78:17: error: ‘PATH_MAX’ undeclared (first use in this function); did
 you mean ‘RAND_MAX’?
    { char fname[PATH_MAX], outname[PATH_MAX];
                 ^~~~~~~~
                 RAND_MAX
```

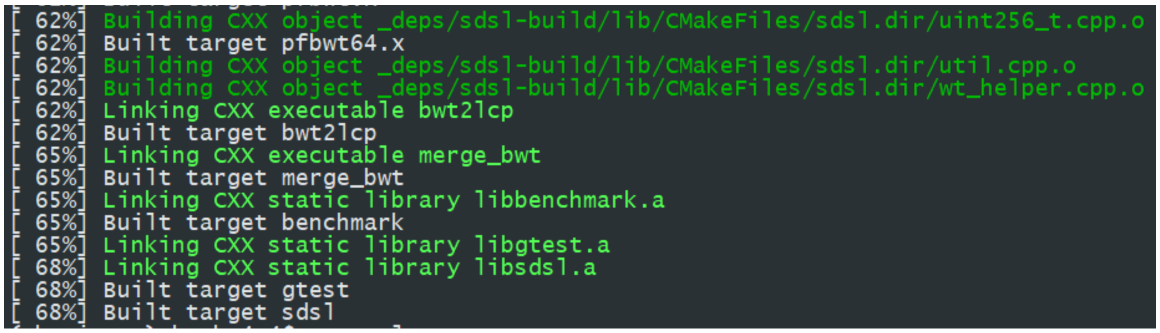
Figure 3.3: A fatal error of undeclared variable while running *cmake*.

As a result, variable *PATH_MAX* was declared in advance in the *makefile* with its maximum possible number as *-DPATH_MAX=4096*.

With the above modifications of *makefile*, the cmake processed successfully. Next, the PHONI can be built using the following *make* command, which is able to keep track of the building process using a log file:

make -j 2>log

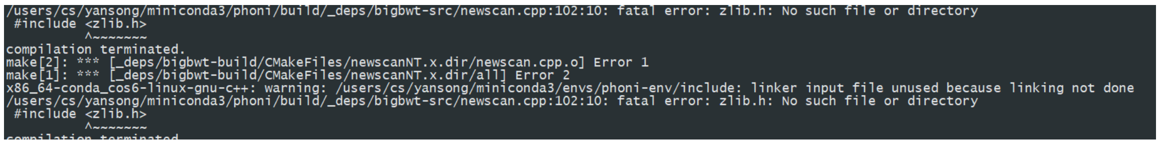
Unfortunately, the *make* process stopped at 68% as Figure 3.4 indicated.



```
[ 62%] Building CXX object _deps/sdsl-build/lib/CMakeFiles/sdsl.dir/uint256_t.cpp.o
[ 62%] Built target pfbwt64.x
[ 62%] Building CXX object _deps/sdsl-build/lib/CMakeFiles/sdsl.dir/util.cpp.o
[ 62%] Building CXX object _deps/sdsl-build/lib/CMakeFiles/sdsl.dir/wt_helper.cpp.o
[ 62%] Linking CXX executable bwt2lcp
[ 62%] Built target bwt2lcp
[ 65%] Linking CXX executable merge_bwt
[ 65%] Built target merge_bwt
[ 65%] Linking CXX static library libbenchmark.a
[ 65%] Built target benchmark
[ 65%] Linking CXX static library libgtest.a
[ 68%] Linking CXX static library libsdsl.a
[ 68%] Built target gtest
[ 68%] Built target sdsl
```

Figure 3.4: *make* processing stopped at 68%.

The reason is that a file named “zlib.h” cannot be located as Figure 3.5 indicated.



```
/users/cs/yansong/miniconda3/phoni/build/_deps/bigbwt-src/newscan.cpp:102:10: fatal error: zlib.h: No such file or directory
 #include <zlib.h>
          ^~~~~~~~
compilation terminated.
make[2]: *** [_deps/bigbwt-build/CMakeFiles/newscanNT.x.dir/newscan.cpp.o] Error 1
make[1]: *** [_deps/bigbwt-build/CMakeFiles/newscanNT.x.dir/all] Error 2
x86_64-conda_cos6-linux-gnu-c++: warning: /users/cs/yansong/miniconda3/envs/phoni-env/include: linker input file unused because linking not done
/users/cs/yansong/miniconda3/phoni/build/_deps/bigbwt-src/newscan.cpp:102:10: fatal error: zlib.h: No such file or directory
 #include <zlib.h>
          ^~~~~~~~
```

Figure 3.5: A fatal error of cannot locate the target file while running *make*.

After several different modifications with respect to the *cmake* file, zlib.h file, and the newscan.cpp file that cannot locate zlib.h includes adding the header file line of zlib.h in that specific file, the fatal error still exists. As a result, the PHONI cannot be successfully built on the Timberlea or Waverley server (two servers of Dalhousie University). The rest of the experiment was conducted by Dr. Massimiliano Rossi on a server of the University of Florida.

3.5 Converting VCF files of chromosome copies to FASTA files.

Before the computation of matching statistics, two VCF files that contain copies of chromosome 19 were converted to FASTA files because of the PHONI only accepts FASTA files as it inputs. For the preparation of the file conversion, the indexes of two VCF files were built using tabix, which is a "generic indexer for TAB-delimited genome position files". The following commands were applied for indexes building.

```
tabix -p vcf chr19.1000.indels.vcf
tabix -p vcf chr19.99.indels.vcf
```

Next, the sequences of chromosome 19 were extracted independently using the *consensus* function of bcftools with the following commands.

```
s_ids_99=(`cat names_99.txt `)
for i in ${$s_1ds_99[@]}; do
bcftools consensus -f chr19.fa -H 1 -s $i -o $i.1.19.fa chr19.99.indels.vcf;
done
s_ids_1000=(`cat names_1000.txt `)
for i in ${$s_1ds_1000[@]}; do
bcftools consensus -f chr19.fa -H 1 -s $i -o $i.1.1000.fa chr19.1000.indels.vcf;
done
```

Finally, two sets of independent sequences can be concatenated into two FASTA files using the following commands.

```
cat $(ls -t) > ../chr19_query.fa
cat $(ls -t) > ../chr19_ref.fa
```

Where the *chr19_query.fa* contains 99 copies of chromosome 19, and the *chr19_ref.fa* contains 1000 copies of chromosome 19.

3.6 Combining founders' files to a single FASTA file.

There were 16 founder sequences generated from the 1000 copies of chromosome 19, each of them was stored independently in a file, which is not an acceptable as the input of PHONI. Hence, all 16 files were converted into one single FASTA file using the following command.

```
mkdir original
mv 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 REF original
founders=(`ls -t original`)
for i in ${founders[@]}; do sed 's/-//g' original/$i > $i; done
touch founders.fa
for i in ${founders[@]}; do echo ">$i" >>founders.fa; cat $i >>founders.fa; echo "" >>
founders.fa; done
```

3.7 Matching statistics computations.

The procedure of computing the two sets of matching statistics can be initiated once the input FASTA files are properly prepared. For the step of the computations, the indexes of the set of founder sequences and 1000 copies of chromosome 19 were built using the following commands.

```
./no_thresholds chr19_ref.fa -f -t 8 -m -s ./test/src/build_phoni chr19_ref.fa
./no_thresholds founders.fa -f -t 8 -m -s ./test/src/build_phoni founders.fa
```

Next, the *chr19_query.fa* was processed using splitpattern.py as the preparation of the query with the following command.

./splitpattern.py chr19_query.fa chr19_query.fa.dir

For the final matching statistics generation, the following commands were applied with the files prepared previously.

./build/test/src/phoni chr19_ref.fa -p chr19_query.fa

./build/test/src/phoni founders.fa -p chr19_query.fa

Finally, two sets of lengths were extracted from the matching statistics for length distribution analyses.

Meanwhile, the pseudo lengths with respect to the 1000 copies of chromosome 19 and its founders are also computed for double-checking the accuracy of the founder sequences.

SECTION 4 RESULTS AND DISCUSSION

4.1 The results of founder sequences generation.

There were 16 founder sequences with similar lengths generated from the 1000 copies of chromosome 19 where each of them contains around 58 million characters of the alphabet {A, T, C, G, N}. Each of them was formed as a series of characters of the alphabet in one single line. Meanwhile, there are also a number of gaps in the founder sequences that are represented by ‘-’, which can be ignored during the matching statistics computations.

The expectation of the set of founder sequences is to keep the features and details of the original set of DNA sequences as much as possible so that the two sets of length distribution of the MS of it would be similar too. Then, the founder sequences can be proved as a replacement of its original DNA sequences as the text in matching statistics computation. Accordingly, the computations of indexing large collections genomes will not be necessary anymore.

4.2 Integrating the lengths of matching statistics into groups.

Two sets of lengths from the two sets of matching statistics were stored into files *chr19_queries.fa_chr19_ref.fa.lengths.bz2* and *chr19queries.fa_founders.fa.lengths.bz2*. There are 5,802,813,825 length values in both sets of matching statistics. Next, the maximum values of two sets of lengths were located using a simple traversal code, which are 2,924,572 characters concerting two copies of chromosome 19 and 2,364,212 characters concerting founder sequences. For the uniformity of the histogram, the lengths of founder sequences were sorted into 237 groups and 293 groups of chromosome 19 using a counting program; therefore, each group contains the length values with a range of 10,000 characters. Moreover, the numbers of lengths in the fiftieth group or later are stable at around 1000,000 characters, which are too small to be considered due to the large total numbers of lengths as shown in Table 4.1 and Table 4.2. Therefore, the first fifty groups of length values are mainly focused.

length value range	<10000	10000+	20000+	30000+	40000+	50000+	60000+	70000+	80000+	90000+
#lengths (chr19s)	1285669463	1101978781	845573673	626136133	454343077	327742166	235431482	169031748	122517381	89484257
length value range	100000+	110000+	120000+	130000+	140000+	150000+	160000+	170000+	180000+	190000+
#lengths (chr19s)	66034976	49198869	36942647	28232653	21808466	17163492	13656575	11037399	9205001	7679349
length value range	200000+	210000+	220000+	230000+	240000+	250000+	260000+	270000+	280000+	290000+
#lengths (chr19s)	6536039	5676656	5069196	4560979	4070628	3530192	3016741	2657244	2420445	2155204
length value range	300000+	310000+	320000+	330000+	340000+	350000+	360000+	370000+	380000+	390000+
#lengths (chr19s)	2001483	1887949	1790795	1744214	1687385	1631596	1615326	1589342	1553072	1549974
length value range	400000+	410000+	420000+	430000+	440000+	450000+	460000+	470000+	480000+	490000+
#lengths (chr19s)	1540000	1526629	1506868	1500000	1490473	1490301	1489105	1471367	1535976	1454976
length value range	500000+	510000+	520000+	530000+	540000+	550000+	560000+	570000+	580000+	590000+
#lengths (chr19s)	1290396	1136022	1047200	1013813	1010000	1010000	1010000	1010000	1010000	1005336

Table 4.1 First sixty groups of length values in the set of chromosome 19

length value range	<10000	10000+	20000+	30000+	40000+	50000+	60000+	70000+	80000+	90000+
#lengths (founders)	5034982107	300806920	102102185	44131903	21119877	10196179	5945978	4372690	3768533	3527181
length value range	100000+	110000+	120000+	130000+	140000+	150000+	160000+	170000+	180000+	190000+
#lengths (founders)	3300803	3239047	3319388	2923442	2880392	2861393	2830924	2830000	2830000	2830000
length value range	200000+	210000+	220000+	230000+	240000+	250000+	260000+	270000+	280000+	290000+
#lengths (founders)	2830000	2830000	2707488	2448434	2157440	2075246	2017825	1991448	1990000	1990000
length value range	300000+	310000+	320000+	330000+	340000+	350000+	360000+	370000+	380000+	390000+
#lengths (founders)	1983542	1980000	1980000	1980000	1980000	1980000	1980000	1980000	1980000	1980000
length value range	400000+	410000+	420000+	430000+	440000+	450000+	460000+	470000+	480000+	490000+
#lengths (founders)	1980000	1980000	1980000	1980000	1980000	1976219	1960000	1668164	1011297	990000
length value range	500000+	510000+	520000+	530000+	540000+	550000+	560000+	570000+	580000+	590000+
#lengths (founders)	990000	990000	990000	990000	990000	990000	990000	990000	990000	990000

Table 4.2: First sixty groups of length values in the set of founders.

Also, the percentage of the number of lengths in the current group with respect to the total number of lengths was calculated for both sets of lengths using Excel as shown in Table 4.3.

length value range	<10000	10000+	20000+	30000+	40000+	50000+	60000+	70000+	80000+	90000+
Founders	86.7679%	5.1838%	1.7595%	0.7605%	0.3640%	0.1757%	0.1025%	0.0754%	0.0649%	0.0608%
Chr19	22.1560%	18.9904%	14.5718%	10.7902%	7.8297%	5.6480%	4.0572%	2.9129%	2.1113%	1.5421%

Table 4.3: First ten groups of percentages of two sets of matching statistics.

4.3 An outstanding number of same amounts of length values (990,000).

As Table 4.2 indicates, the number of length values tends to stabilize at 990,000 characters after the 50th group in the set of founder sequences. Similarly, the set of 1000 copies of chromosome 19 also demonstrate such a changeless trend from the 64th group. Such a number remains the same until the 233rd group, which takes 77.2% and 57.7% of the total number of groups respectively of the set of founder sequences and chromosome 19. A hypothesis is that the range of the groups that share the same amount of length values depends on the range of the MEMs. It stands for maximum exact matches, and "an exact match of string S_1 and string S_2 is a substring of length l which occurrence starts at position p_1 in S_1 and at position p_2 in S_2." (Ohlebusch, 2010). Moreover, the exact match will be a maximum exact match if it satisfies the left maximality and right maximality simultaneously, which are "p_1 = 1 or p_2 = 1 or $S_1[p_1 - 1] \neq S_2[p_2 - 1]$" and "$p_1 = n_1$ or $p_2 = n_2$ or $S_1[p_1 - 1] \neq S_2[p_2 - 1]$" mathematically where n_1 and n_1 is the length of S_1 and S_2 respectively. Besides, the exact number of the length values depends on the number of references in the input pattern and the range of characters in a group of length values.

More precisely, there are 99 copies of chromosome 19 formed as the input pattern of the matching statistics and the range of characters in each group is 10,000 characters. For the set of founder sequences, the groups that share the same amount of length values are from the 50th to the 233rd, which means each of the 99 patterns chromosome 19s has a MEM with a length of 2,340,000 characters and no other MEMs has a length longer than 499,999 characters. Similarly, each chromosome 19 in the 99 patterns of the set of chromosomes 19 would have a MEM with a length of 2,340,000 characters and no other

MEMs has a length longer than 639,999 characters. Furthermore, the number of length values within such two ranges depend on the number of the patterns and the number of characters in each group. Each length in each group with such ranges will appear once for each copy of chromosome 19 in the patterns, so the number of lengths in each group will be the size of the group multiplied by the number of pattern chromosomes. Therefore, the constant number of length values 990,000 in each group is the result of 99 pattern chromosomes multiplying 10,000 characters in each group. Besides, if there are two MEMs much longer than others for each copy of chromosome 19 in the patterns, then there will be a massive number of groups that have twice the amount of length values compare to one MEM, which is 1,980,000. Such a hypothesis is only based on one single dataset and its founder sequences, which means it requires further research for more certainty.

4.4 Meaningful range of groups analysis in two set of length values.

The length values of the set of founder sequences were divided into 237 groups and 293 groups for the set of chromosome 19, which is too broad to be showed in histograms; hence, a meaningful range of groups have to be decided before the histogram analysis. There were more than 11.6 billion length values collected in total, each set of matching statistics used 99 copies of chromosome 19 as the pattern during the computation; hence each copy has around 58 million matched longest prefixes with respect to the characters of the copies of chromosome 19. All the length values were sorted into groups of 10,000 characters based on the maximum number in the set.

The most striking value in the entire dataset is the number of length values that are under 10,000 characters in the set of founder sequence 5,034,982,107, which is more than

86% of the total. Such an enormous amount of short matches indicates the correlation between the 99 copies of chromosome 19 and the set of founder sequences is not strong in the first place. Meanwhile, the amount of the same group in the set of chromosome 19 just takes less than 23 % of the total, which shows a better relevance between the 99 copies of chromosome 19 and the 1000 copies of chromosome 19.

Since the total number of length values in each set is more than 5.8 billion, the groups that have the number of lengths with 1000,000 or smaller only takes 0.0171% of the total, which is not capable of affecting the distribution of lengths in the set. More specifically, the number of length values of the 49th group in the set of founder sequences is 990,000, and the rest of the groups remain the same amount or lower; hence, the 49th to last group of length values in the set of founder sequences do not affect the entire length distribution very much. Similarly, the numbers of lengths from the 63rd group to the last group in the set of chromosome 19 are also equating to 990,000 or lower than it. Therefore, from 49th to the last group of two sets of matching statistics can be neglected in histogram analysis of the next stage.

4.5 Histograms based on the lengths shorter than 500,000 characters.

Based on the meaningful range conclusion from section 4.3, a histogram was constructed from the first fifty groups of length values of two sets of matching statistics as shown in Figure 4.1.

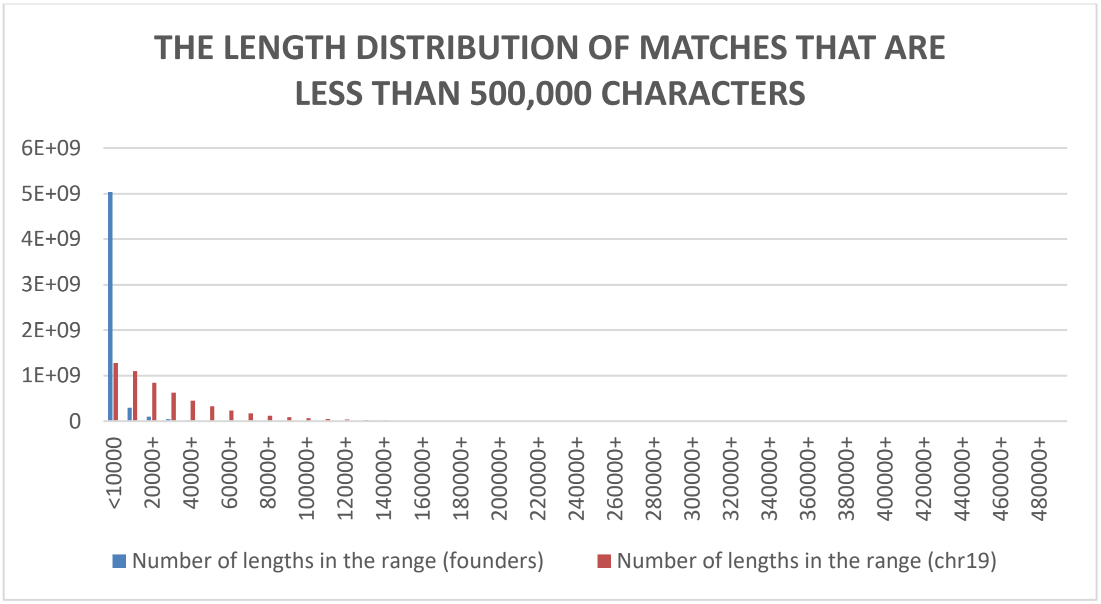


Figure 4.1: Distributions of lengths that are less than 500,000 characters.

Generally speaking, two sets of length values both show a falling trend for the range of 0 to 500,000 characters. However, the histogram also clearly indicates that the numbers of length values after the 15th groups are too small to be illustrated in such a range of values; hence, the 16th to 49th groups of length values need to be separated to visualize the distributions of them. Therefore, a sub-range histogram was built based on the length values in the range of 16th to 49th groups as shown in Figure 4.2.

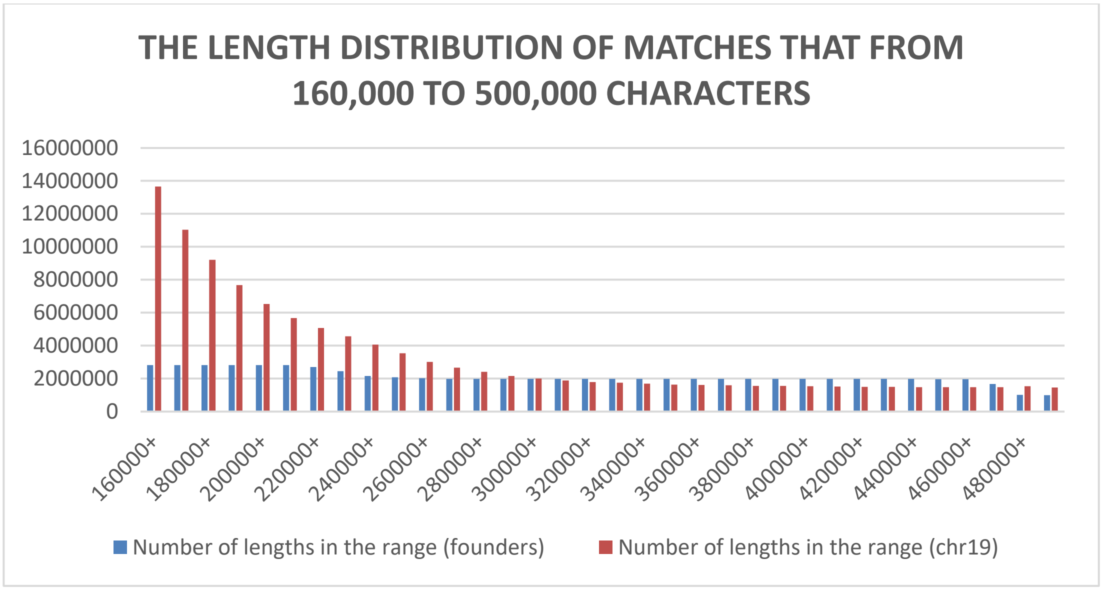


Figure 4.2: Distributions of lengths within the range of 160,000 to 500,000 characters.

As Figure 4.2 indicates, the length values of chromosome 19 do demonstrate a decreasing trend in the range of 160,000 to 500,000 characters. Nevertheless, the length values of founder sequences are almost evenly distributed in the same range of characters, which is the opposite of the changing pattern shown by the two sets of matching statistics in the original range of 0 to 500,000 characters. More specifically, such an even length distribution of the set of founder sequence indicates that the numbers of length values quickly flattened out after experiencing a sharp decline in the first fifteen groups, but the less dramatic decrease in another set continues after the first fifteen groups. Moreover, the numbers of length values in these groups are not as big as the numbers of the first fifteen groups. Therefore, such a range of groups of lengths is not a critical component for the length distributions comparison of the two sets of matching statistics. Thus, the first fifteen groups of length values were more influential for the length distributions concerning the two sets of matching statistics.

4.6 Histogram based on the lengths shorter than 160,000 characters.

The length values in the range of 0 to 160,000 characters take up to over 94% of the total number of lengths in the set of chromosome 19 and over 95% in the set of founder sequences, which means the length distributions of the two sets of matching statistics are mainly determined by these 16 groups of length values. Therefore, a histogram was built based on the length values that within this range for simplicity and clarity as shown in Figure 4.3.

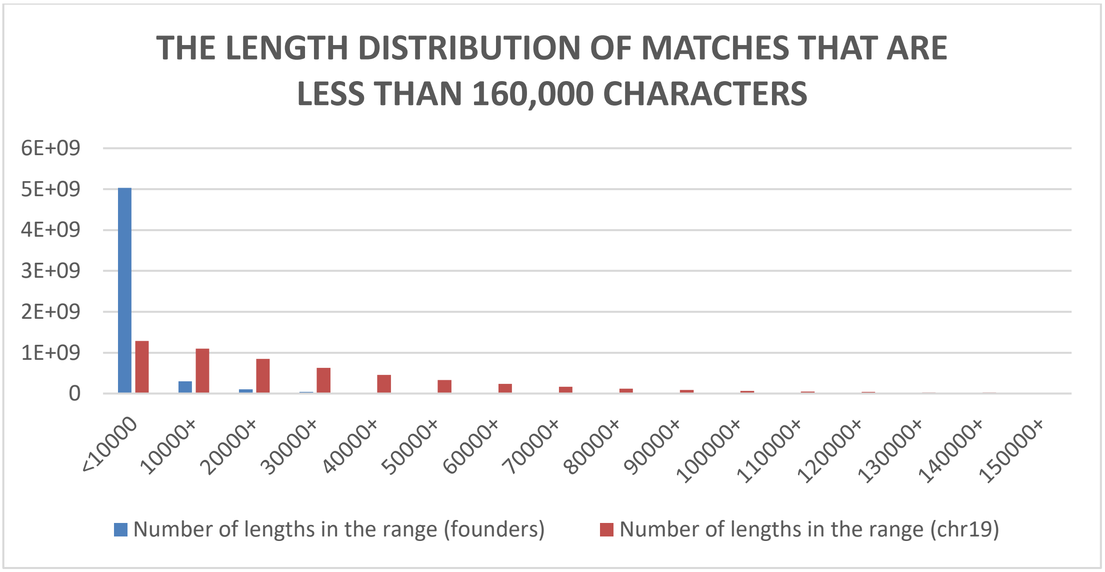


Figure 4.3: Distributions of lengths that are less than 160,000 characters.

This histogram demonstrates a similar length distribution with the one shown in Figure 4.1, which means the hypothesis of ignored 16th to 49th groups do not affect the length distribution is correct. For the length distribution show in Figure 4.3, the group of fewer than 10,000 characters occupies around 5 billion length values in the set of founder sequences, then it decreases significantly after this group. Furthermore, the groups of length values from 10,000 to 40,000 characters are still visible on the histogram even after the dramatic decline, but the groups that contain length values of 40,000 characters or larger are too small to be noticed. Therefore, the matching statistics of founder sequences are not as positive as the expectation. On the other hand, the set of chromosome 19 shows a more gradual decline in the same range, and it is visible until the group of 130,000 characters. Therefore, the difference between the two sets of length distributions is not as small as excepted since the set of founder sequence has a significant amount of shorter length matches than the set of chromosome 19. For a better analysis of the distributions, the length values in the group of less than 10,000 characters will be further investigated for both sets of the matching statistics.

4.7 Histogram based on the lengths shorter than 10,000 characters.

Moreover, the extreme small length values that less than 10,000 characters take up to over 86% of the total lengths in the set of founder sequences, a more detailed analysis of the length values in the two sets of matching statistics that are less than 10,000 characters were conducted using a histogram as Figure 4.3 shown.

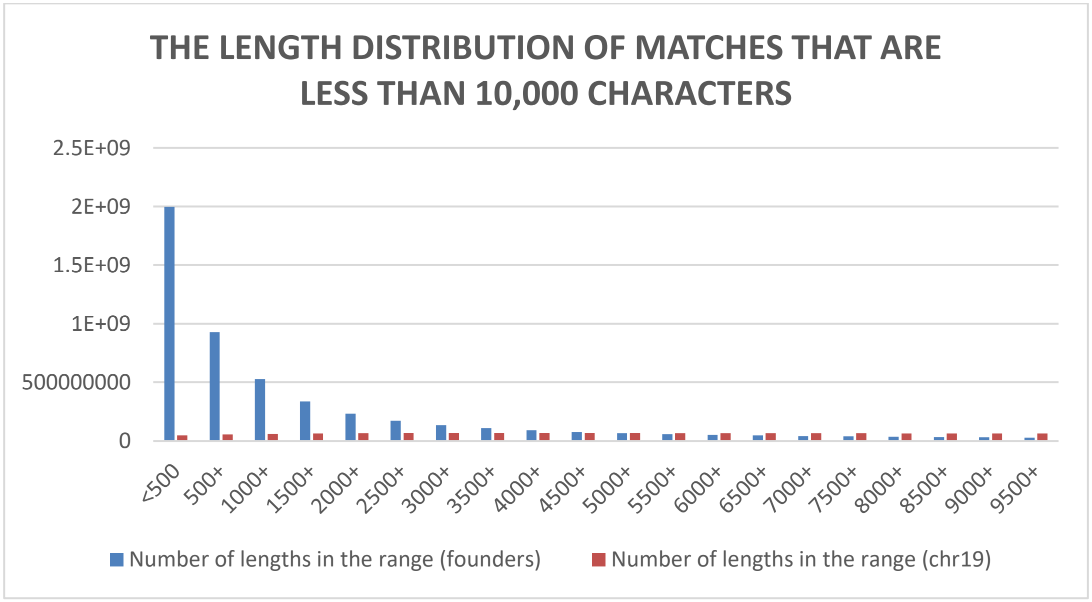
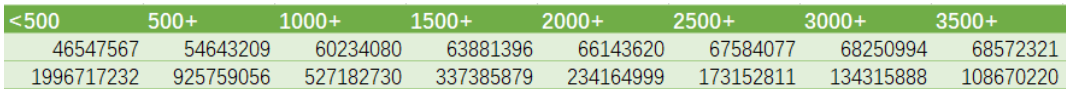


Figure 4.4: Distributions of lengths that are less than 10,000 characters.

The length distribution of matches that are less than 10,000 characters is rather similar to the distribution of the range of 160,000 to 500,000 that shown in Figure 4.2 but conversely. The length values in the set of founder sequences still perform an obvious decrease in the range, but the rate of decline has slowed. According to the histogram, there are still almost 2 billion matched longest prefixes that have a length of fewer than 500 characters, which is around 34.4% of the total number of matches in the set of founder sequences. Also, the amounts of lengths values in the range of 500 to 4000 are noticeable compare to the same groups in the set of chromosome 19 as Table 4.4 shown.



<500	500+	1000+	1500+	2000+	2500+	3000+	3500+
46547567	54643209	60234080	63881396	66143620	67584077	68250994	68572321
1996717232	925759056	527182730	337385879	234164999	173152811	134315888	108670220

Table 4.4: The number of lengths in the range of 0 to 4000 characters.

Due to the larger amounts of length values, all the groups of length values are visible in Figure 4.4, which means the differences between the largest and the smallest number of length values in this range are also narrowed down. Therefore, the length values in the set of founder sequences are indeed more compactly distributed in the range of 0 to 10,000 characters, but the number of length values in the group of fewer than 500 characters still accounts for the vast majority of matches. For the set of chromosome 19, the amount of the length values in each group is equally distributed in this range, which indicates the total number of matches with length less than 10,000 characters is too small to show a change of trend. More specifically, there are 1,285,793,593 length values in the range of 0 to 10,000 characters, which is only about a quote of the number of length values in the founder sequences set. Under such circumstances, the two sets of length distributions can be concluded that the difference between them is significant, the set of founder sequences has a much shorter length especially in the range of 0 to 500 characters. Further analysis can be conducted in such range for both sets of matching statistics, but the length distribution of chromosome 19 could maintain the same status, which is evenly distributed across the entire range.

4.8 Histogram based on the lengths shorter than 500 characters.

For a more precise judgement with regard to the distribution of two sets of matching statistics, the length values that less than 500 characters was divided into 10 groups and a more detail histogram was built as the Figure 4.5 shown.

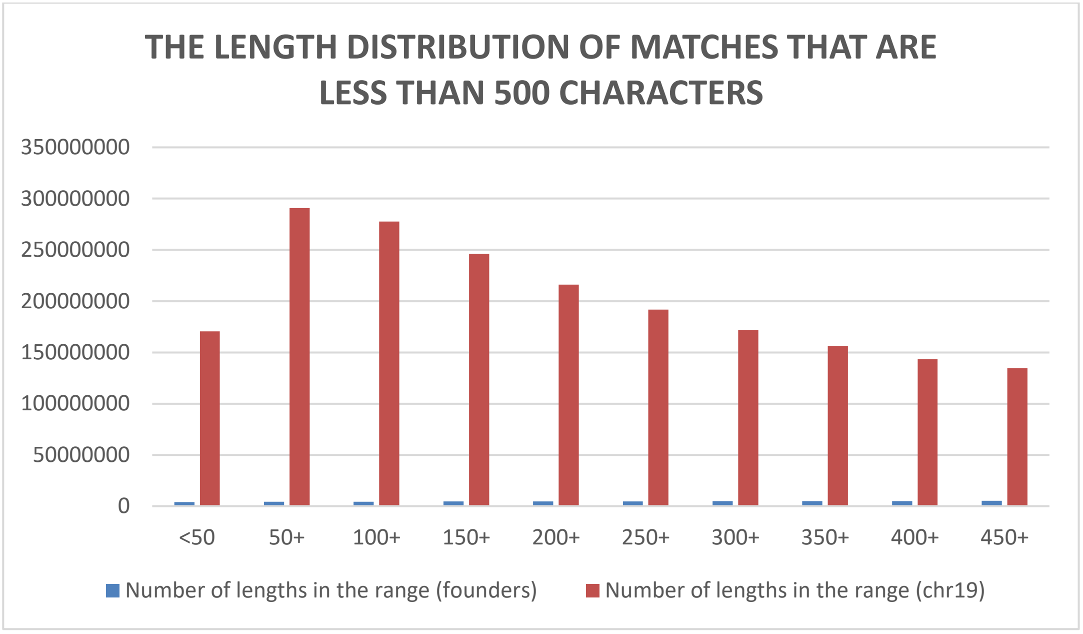


Figure 4.5: Length distributions for values that are less than 500 characters.

As the previous section expected, the length values in each group of the chromosome 19 set do equally distributed in the range of 0 to 500 characters. However, the length values in the set of founder sequences rather demonstrate a different trend than before. The first group that contains length values less than 50 characters is not the biggest number anymore. There is an increase of 70% from the first group to the second group. Then, a decreasing trend still exists, but it starts from the second group, which contains the length values from 50 to 100 characters. Meanwhile, the rate of descent slows down notably, which only drops 21% from the second group to the tenth group. For the comparison of two sets of length distributions, the diversity is extraordinary in terms of quantity, the number of length values in the set of chromosome 19 is only 1 to 2% of the number of lengths values in the set of founder sequences. From the perspective of trends, there is still a discrepancy between two sets of matching statistics, but it is definitely narrowed down. Overall, the set of chromosome 19 does not distribute enough amount of matches in the range of length that can alter the entire length distribution. For the set of founder sequences, a decreasing trend still stands as same as the previous ranges of histograms if the group of length values less than 50 characters is ignored.

The analysis of the length distributions of two sets of matching statistics is conducted based on four ranges of length values, which are less than 500,000 characters, less than 150,000 characters, less than 10,000 characters, and less than 500 characters. From the first two histograms with length values up to 500,000 and 150,000 characters, both sets matching statistics show a decreasing trend, but the slope of the set of founder sequences is much bigger. Hence, the number of short matches in the set of founder sequences has exceeded the expectation of the experiment. From the third histogram that includes length values up to 10,000 characters, the set of founder sequence maintained the previous sharp decline, but the other set of chromosome 19 showed an even length distribution. In this range, the number of matches in the set of chromosome 19 does not have a decisive influence on its overall length distribution, but the number of matches in a set of founders is still large enough to show a decreasing trend. The fourth histogram only includes the length values up to 500 characters, which is the narrowest range. Moreover, both sets of matching statistics did not show a rapid drop, and the number of matches demonstrated a significant gap between the two sets. Therefore, the set of chromosome 19 does not have a number of matches distributed in this range. For the other set, the total number of matches is still significant, but the differences between each group are relevantly small in this range.

Overall, the majority of length values are under 40,000 characters in the set of chromosome 19; however, the set of founder sequences are mainly distributed in the range of 0 to 3000 characters. Therefore, much shorter matches in the set of founder sequences appear more frequently than expected compare to the set of chromosome 19. Such a considerable difference of length distributions between two sets of matching statistics reveals the set of founder sequences cannot be a proper replacement of its original set of DNA sequences as the input text for matching statistics computation. Under current circumstances, the necessity of indexing large collections of genomes has been proved.

4.9 Pseudo length distributions analysis using histograms.

The analyses of length distribution with respect to the 1000 chromosome 19s and it founder sequences indicate that the founders are not accurate enough to replace the set of original DNA sequences for indexing. For supporting such a conclusion, two sets of pseudo lengths were also computed from 1000 chromosome 19s and its founder sequences respectively. The pseudo lengths are computed using BWT and thresholds rather than PHONI, which means they are not matching statistics. The algorithm is simple. While backward searching the pattern, If the BWT character is matched then increase the pseudo length by 1, if a threshold is used then reset the pseudo length to 0.

At every position, write the pseudo length. With the experience from previous sections, there are three histograms that can illustrate the length distributions effectively, which are histograms with less than 500,000 characters, 160,000 characters, and 10,000 characters.

4.9.1 Histogram based on the pseudo lengths shorter than 500,000 characters.

The first histogram gives a general view of the two sets of pseudo lengths in case there are long matches distributing in large number groups. As Figure 4.6 indicated, similar to the length distributions, the pseudo lengths of the founder sequences also demonstrate a rapid decrease after the group of fewer than 10,000 characters, and the pseudo length from the set of 1000 chromosome 19s gives a more gradual decline compares to the set of founders. Hence, the set of founder sequences is still not a satisfying replacement of its original set of DNA sequences when it comes to indexing them.

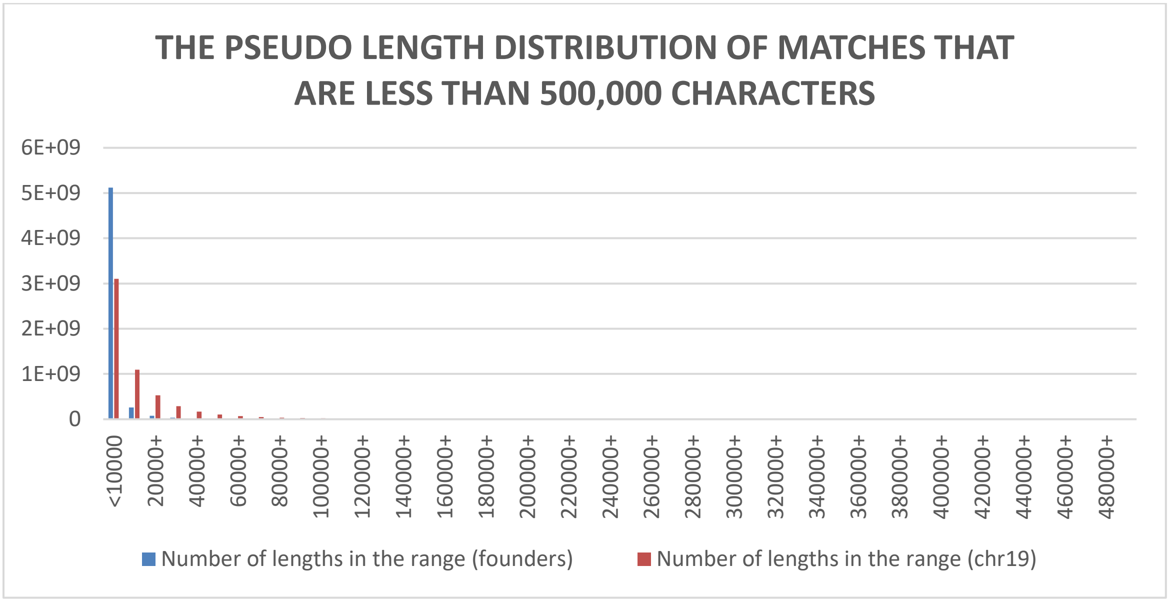


Figure 4.6: Distributions of pseudo lengths that are less than 500,000 characters.

For the set of founder sequences, groups that have pseudo lengths longer than 40,000 characters are not visible from the graph. Meanwhile, the groups of the 1000 chromosome 19s that contain pseudo length values that are longer than 100,000 characters are also too small to be visible on the histogram. Therefore, further analysis should base on a narrower range of length values.

4.9.2 Histogram based on the pseudo lengths shorter than 160,000 characters.

The range of pseudo length values was narrowed down to 160,000 characters in Figure 4.7, which gives a clearer view of the groups that determine the pseudo length distributions. Compare to the length distributions from two sets of matching statistics, the pseudo length distributed very similarly; except the gap between the two groups that contain pseudo lengths less than 10,000 characters is much smaller, which drops from 3,749,312,644 to 2,014,919,142. Such a smaller gap indicates the correlation between the set of chromosome 19s and its founders is stronger from the perspective of pseudo lengths. However, the gap still takes 34.7% of the total number of pseudo values, which means the founder sequences still has much shorter matches compare to the chromosome 19s.

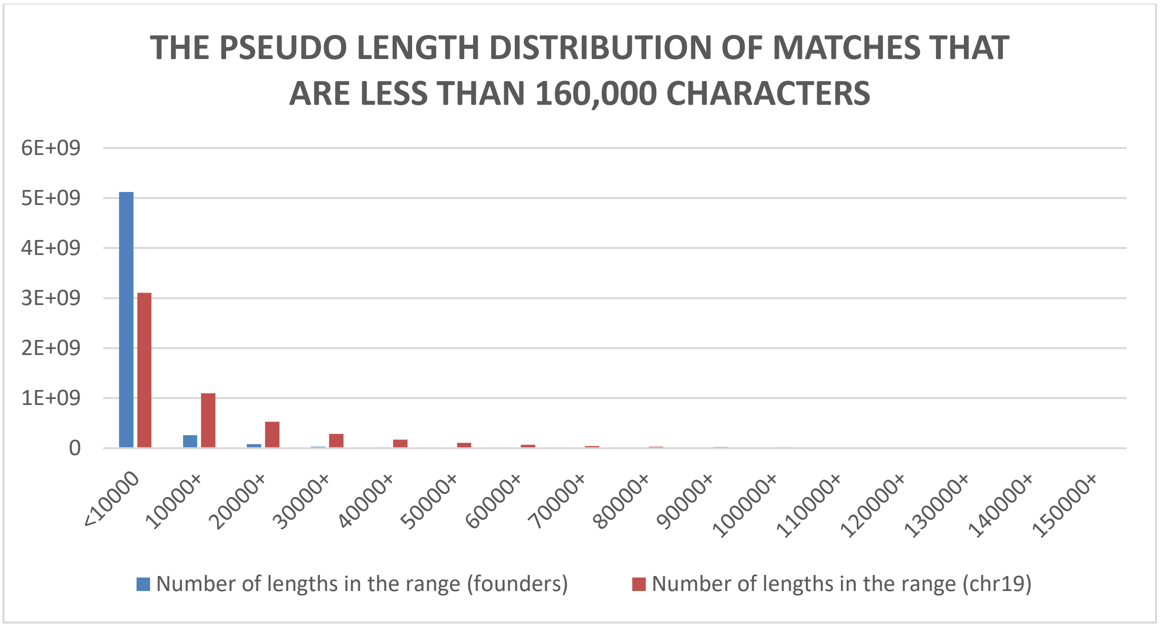


Figure 4.7: Distributions of pseudo lengths that are less than 160,000 characters.

4.9.3 Histogram based on the pseudo lengths shorter than 10,000 characters.

The last step of pseudo length analysis was based on the matches that are less than 10,000 characters as Figure 4.8. The pseudo length distribution within this range is still distributed very similar to the length distribution. More specifically, the pseudo lengths of chromosome 19s are almost equally distributed in each group; however, the pseudo lengths of the founders are still mainly distributed in a group that has the shortest matches, which are the group of length values that are less than 500 characters. The only minor difference between pseudo length distribution and length distribution within this range is that there are more pseudo length values in each group compare to length values because the total number of pseudo lengths is larger than the number of lengths as previously indicated.

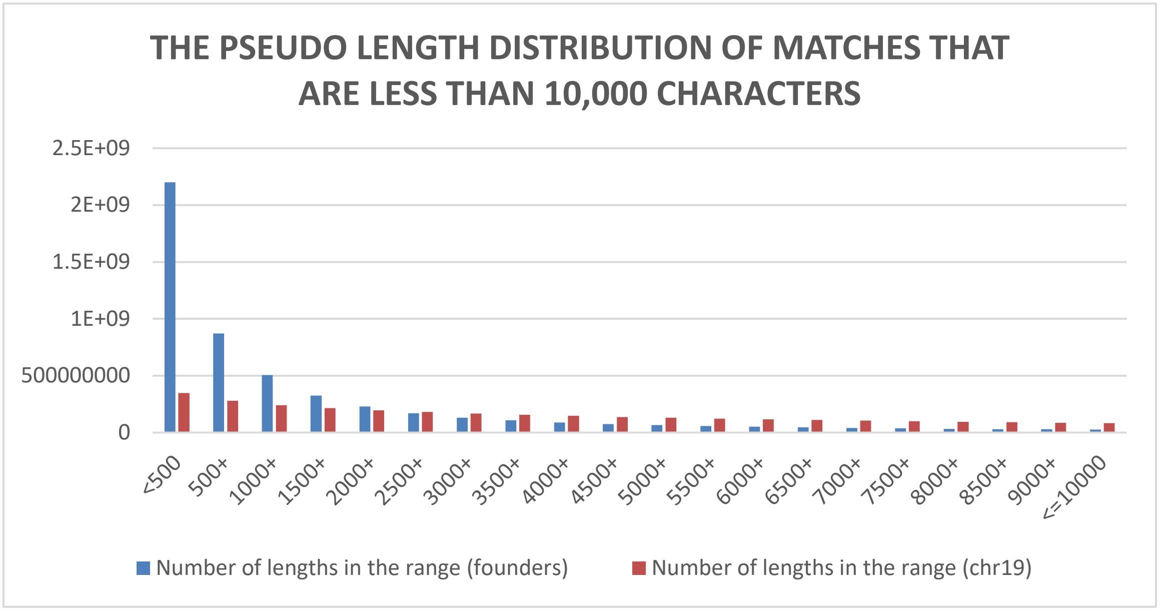


Figure 4.8: Distributions of pseudo lengths that are less than 10,000 characters.

Due to the pseudo length distribution is highly identical with the length distribution, which means the pseudo lengths of the founder sequences are also much shorter than the 1000 copies of chromosome 19. Therefore, the theory that a set of founder sequences cannot be a proper replacement of its original DNA sequences for matching statistics computations under the circumstances of the experiment.

SECTION 5 CONCLUSION AND FUTURE WORK

The overall goal of this research is to determine whether a set of founder sequences can replace its original set of DNA sequences as the text in matching statistics computations. Based on a quantitative and qualitative analysis of the length distributions of two sets of matching statistics, it can be concluded that the founder sequences cannot be a replacement of its original set of DNA sequences as the text in matching statistics computations. Therefore, indexing large collections of genomes in matching statistics is still an indispensable procedure under the circumstances of this experiment.

In general, the procedure of this research was composed of three stages, which were founder sequences generation, matching statistics computation, and histogram production. There were 16 founder sequences generated from the 1000 copies of chromosome 19 as the preparation of matching statistics computation in the first stage. Then two sets of matching statistics were computed using PHONI and the length values were stored in stage two. At the last stage, two histograms were generated from the two sets of length values for determining whether the set of founder sequences can be the replacement of the 1000 copies of chromosome 19 in matching statistics computation.

The results of the experiment indicate that the length of the matches in the set of matching statistics generated from 99 copies of chromosome 19 concerning the set of founder sequences are dramatically shorter than the set of 1000 copies of chromosome 19. The lengths values that under 10000 characters occupied the majority of matches in the set of founder sequences, but there are only less than a quarter of such matches in the set of chromosome 19. From the perspective of the global length distribution, the set of founder sequences presents a sudden decrease after when the length values exceed 10000 characters; whereas the length distribution in the set of chromosome 19 shows a gradual downward trend as the length values increase. More precisely, the length values are equally distributed within the range of 10000 characters in the set of chromosome 19; however, it still emerges a significant decrease in the range of 0 to 4000 characters in the set of founder sequences. Besides, the pseudo lengths of the chromosome 19s and its founders are also demonstrating very similar distributions with the lengths. Therefore, a set of founder sequences is not qualified to be the alternative of its original set of DNA sequences in matching statistics computations.

The research was based on 1099 copies of chromosome 19 from the database of the 1000 Genome Project. It demonstrated the differences in the length distributions between the set of founder sequences and the references of chromosome 19, which reflected the essentialness of indexing large collections of DNA sequences . However, such a conclusion has its limitations since the experiment was conducted from a partial database that only contains a single type of chromosome, which lacks the diversity of data. Therefore, the results of the experiment could be different if the matching statistics are computed based on an entire genome database with various types of chromosomes.

The future research of the using founder sequences as the text of matching statistics would be mainly concentrated on replacing the 1000 references of chromosome 19 with a real genome database for the matching statistics computations rather than using just a single type of chromosome as the input text and the base of founder sequences generation. Meanwhile, the future research will still aim for confirming whether the huge difference in the length distribution still exists between the two sets of matching statistics with respect to a database of genomes and its founder sequences. Besides, more diverse and detailed investigations of the length distribution in matching statistics will be conducted. From the results of the current experiment, the necessity of indexing large collections of genomes is temporarily unchangeable, but eventually, there will be a more effective alternative for such a task along with the development of technology.

BIBLIOGRAPHY

Boucher, C., Gagie, T., I, T., Köppl, D., Langmead, B., Manzini, G., Rossi, M. (2021, February 11). *PHONI: Streamed matching statistics with Multi-Genome References*. Retrieved April 2, 2021, from https://arxiv.org/abs/2011.05610

Higgins, D., Sievers F., Dineen D., Wilm, A. (2021, March 3). *Clustal Omega*. Conway Institute UCD Dublin. http://clustal.org/omega/

Gagie, T. (2020). Dalhousie University. *MONI: Matching Statistics with Multi-Genome References.* Retrieved April 2, 2020, from https://www.dropbox.com/s/06mbaj4w0l16wv1/Travis_Bicocca_lecture_1.mp4?dl=0

Green, C. (2016, October 20). *Cross Entropy*. Heliosphan. Retrieved April 2, 2021, from https://heliosphan.org/cross-entropy.html

IGSR: The International Genome Sample Resource. (2021). *Supporting open human variation data.* Retrieved April 2, 2021, from https://www.internationalgenome.org/data/

Langmead, B. (2020a). *Fighting reference bias with pangenomes*. Video. CeBIB Workshop (November 23, 2020), John Hopkins University. Retrieved April 2, 2021.

Langmead, B. (2020b). Indexing. Retrieved April 2, 2021, from https://www.youtube.com/playlist?list=PL2mpR0RYFQsADmYpW2YWBrXJZ_6EL_3nu

Li, H. (2020). *Bcftools.* GitHub. Retrieved April 02, 2021, from http://samtools.github.io/bcftools/bcftools.html

Mäkinen, V., Belazzougui, D., Cunial, F., & Tomescu, A. (2015). *Genome-Scale Algorithm Design: Biological Sequence Analysis in the Era of High-Throughput Sequencing* (pp. 244-245). Cambridge: Cambridge University Press. doi:10.1017/CBO9781139940023

Norri, T. (2019). *Vcf2multialign.* GitHub. Retrieved April 02, 2021, from https://github.com/tsnorri/vcf2multialign/tree/separate-vcf-parser

Norri, T., Cazaux, B., Kosolobov, D., Mäkinen, V. (2019). *Linear time minimum segmentation enables scalable founder reconstruction.* Algorithms Mol. Biol. 14(1): 12:1-12:15. Earlier in WABI 2018. Retrieved April 2, 2021, from https://www2.helsinki.fi/en/researchgroups/genome-scale-algorithmics/founders

Ohlebusch, E., Gog, S., Kugel A. (2010, October 13). *Computing Matching Statistics and Maximal Exact Matches on Full-Text Indexes.* Institute of Theoretical Computer Science Ulm University Retrieved April 2, 2021, from https://www.uni-ulm.de/fileadmin/website_uni_ulm/iui.inst.190/Mitarbeiter/gog/talks/spire_2010.pdf

Sam tools. (2021, March 21). *Tabix*. Retrieved April 2, 2021, from http://www.htslib.org/doc/tabix.html

Sherman, R.M., Salzberg, S.L. (2020). Pan-genomics in the human genome era. *Nat Rev Genet* 21, 243–254 . https://doi.org/10.1038/s41576-020-0210-7

Ukkonen E. (2002) *Finding Founder Sequences from a Set of Recombinants.* WABI 2002. Lecture Notes in Computer Science, vol 2452. https://doi.org/10.1007/3-540-45784-4_21