IDENTIFYING AND CATEGORISING SINGLE POINT MUTATIONS OF TUMOUR CELLS IN WHOLE HUMAN GENOME

Supervisor:

Dr. Paskin Cherniavski Anat

Presented by:

Chen Asaraf, 203867999
Uri Hanunov, 204558399
Yarden Ben-Amitai, 312575384

Final Project

Department of Computer Science, Ariel University

October, 2020

Abstract

Our research revolves around the field of Targeted therapy, a practice that has been regarded as the biggest success in the treatment of cancer in the past few decades. In this paper, we review our development and results for constructing a feasibly running algorithm for locating, mapping and analysing mutations under the category of Single-point mutation, in a whole human genome. The algorithm we constructed, provides an individual numerical evaluation of mutations and identification of precise changes in comparison to a non-tumour genome. Such information, if implemented correctly, can lead to better detection, diagnosis and treatment provided by healthcare systems. In addition, the methods we describe can be applied to implementations for locating other categories of mutations, thus laying a foundation for advancement.

Keywords: point mutation, whole human genome, de-novo assembly, de-bruijn assembler, edit-distance.

INTRODUCTION

Targeted therapy was devised as a result of the connection found between mutations in the genome and their implications on the best suited treatment for each patient individually[1]. Having discovered that, scientists have begun devising and testing a variety of tools meant to scan, map and analyse DNA sequences to a usable degree.

The process begins, naturally, with sampling DNA from a tissue material, that is a process of determining the order of **nucleotides** in DNA. The four bases are components within a nucleotide and consist of Adenine (A), Thymine (T), Guanine (G) and Cytosine (C), where A--T, G--C are bound together in a bond called **base pair**. The results of such a process, known as **sequencing**, is a collection of unorganised, partially overlapping **reads**, these are fragments from a longer DNA sequence, and can be performed by a substantial number of techniques and methods.

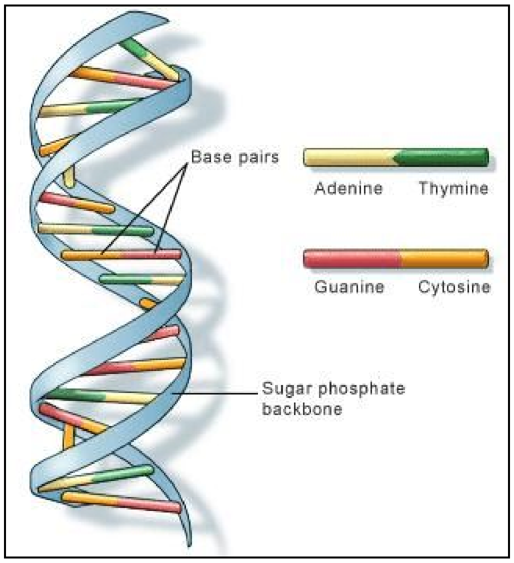

Fig. 1: DNA construct by pairs A--T and G--C

Every such collection of reads has a **coverage** (also, **cover number**), which is the average number of reads that sample a single nucleotide, and can be calculated as such:

$$Coverage = \frac{(Number\ of\ Reads) * Reads_{length}}{Genome_{length}}$$

These reads are aligned and merged with the help of various unique algorithms, in order to reconstruct the original sequence, in a vast and intricate process referred to as **sequence assembly**. This process holds different approaches that we review in the following paragraph.

The length of the human genome is over 3 billion *bp* (base pair), with hundreds of gaps in the sequence and an uncertainty of about 5–10%, therefore, any method used to assemble the genome from fragments, inevitably, must tackle memory and complexity issues. Sequence assembly can be distinguished to two different types: **mapping** [2] , that is assembling reads against an existing backbone sequence, known as the **reference genome** of a species, and **de-novo assembly** [3], which consists of assembling short overlapping reads to create full-length sequences. Mapping results in an assembled genome that is similar but not necessarily identical to the specimen, due to the alignment process with the existing reference. De-novo assembly, on the other hand, assembles the genome in such a way that the uniqueness of the specimen's DNA is preserved, and can even maintain the existing mutations in the final results.

De-novo assemblers operate on top of a **De Bruijn graph**, a data structure of a directed multi-graph, where each vertex is a unique subsequence of length k, referred to as **k-mer**, contained within a read, and every directed edge between vertices x and y implies that some k-mer has prefix x and suffix y. To find a complete genome sequence it is necessary to visit every edge in the graph exactly once, also known, in graph theory, as the **Eulerian cycle**. As we visit every edge in the graph, which represents all possible k-mers, we produce a candidate genome.

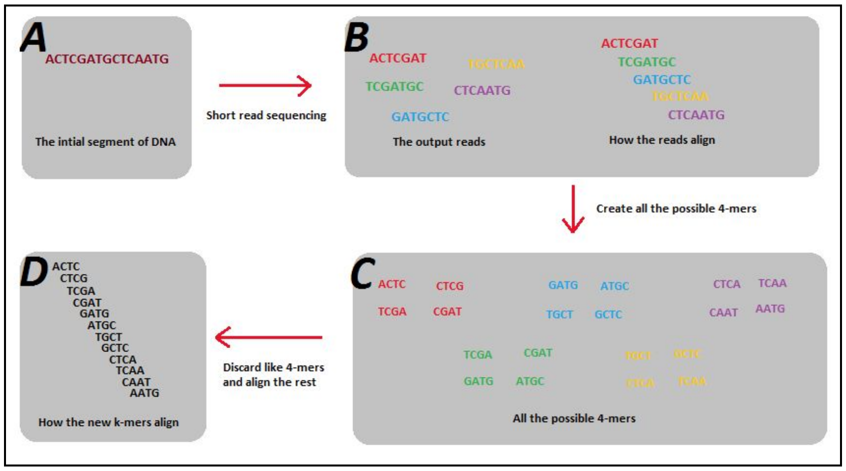

Fig. 2: Flow chart of assembling reads through alignment

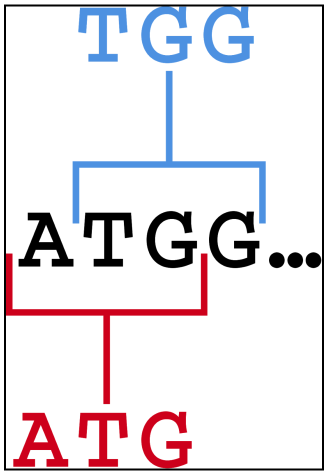

Fig. 3: Two 3-mers of a sequence

However, even when assembled, the complete sequence would be meaningless without a way to interpret it. The next phase is to locate and decipher the different mutations found in the sequence with regard to their quantity, nature and categories [4], all are beneficial parameters that assist in producing quality evaluation of the patient.

A mutation is, simply put, a change in the sequence of bases in the DNA, that can be associated to one of the following: **Germline mutations** occur in gametes (organism's reproductive cells) and therefore can be transmitted to offspring. **Somatic mutations** occur in other cells of the body and are confined to just one cell and its daughter cells and cannot be passed onto an offspring. **Chromosomal alterations** are mutations that change chromosome structure and occurs when a section of a chromosome breaks off and rejoins incorrectly or does not rejoin at all. **Frameshift mutations** are deletion or insertion of one or more nucleotides that changes the reading frame of the base sequence, and lastly, a **Point mutation** occurs when one single nucleotide base is added, deleted or changed in a sequence. The difference between mutation is crucial to an individual's treatment due to the implications each category presents, for example, a human chromosomal alteration is the mutation that causes Down Syndrome, and a frameshift mutation can have a drastic effect on the protein product.

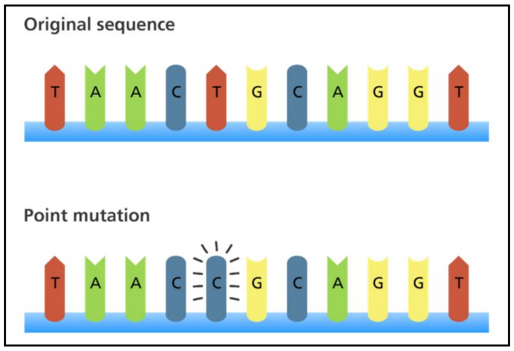

Fig. 4: Point mutation replacement

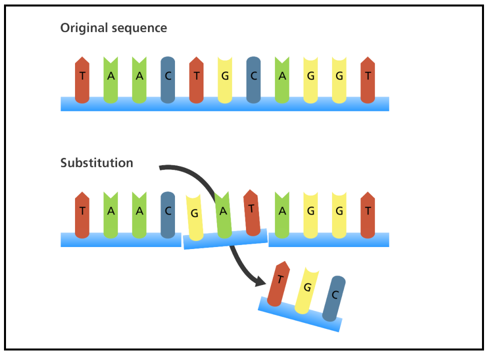

Fig. 5: Frameshift mutation

Unfortunately, as of today, to the best of our knowledge, there are no tools in use that perform the entirety of the process, with the individual's genome as the base for comparison, instead of a reference. In addition, we constructed a foundation for categorising all types of mutation, with emphasis on single point mutation that are often overlooked. Our research lays the foundation work as a means to improve the existing methods for mutation analysis to perform in finite time and acceptable memory usage.

METHODS:

In the following section we present the different processes and techniques that were tested and tuned over the data.

1. INPUT REVIEW:

   Our data originates from two different tissues, taken from the same individual, where the first is a normal tissue and the other is a tumour. Each tissue results in two files, R1 and R2, due to each tissue being sequenced on both ends of a fragment, referred to as **paired-end sequencing** [5], with an average cover number of 100. These data samples are the results of **Illumina NGS** (Next Generation DNA sequencer) dataset of paired-end 100-base reads, and encapsulated in FASTQ file format named *sample_number-T_R1* for the tumour tissue and *sample_number-N_R1* for the non-tumour one, in an overall size of 158GB. Although, ultimately, we did not exploit the pair-end factor, we believe its usefulness in generating high-quality, alignable sequence data can be harnessed.

2. PREPROCESSING THROUGH ASSEMBLING:

Many of the methods we apply were as a result of repetitive testing and tuning that lead us to the importance of preparing the input beforehand. Ultimately, we expect to enhance efficiency and enable future analysis of mutations throughout the entire genome via mapping mutation quantity per chromosome. Initially, the data could have been used with no preprocessing what so ever; nevertheless, we negated this approach for 2 reasons: firstly, while taking into account that there are over 700,000,000 reads, that will inevitably strain the memory and complexity of the algorithm. Secondly, with coverage being so high, it is clear that an unprocessed dataset would contain a large amount of repetitive, redundant information, thus if we were to compare between all reads, it would result in an inaccurate statistic. Lastly, through the connections we create between reads and their placement, we hope to enable future advancement of inferring mutations to specific areas in the genome.

When considering possible assembly programs to implement in our algorithm we had to decide between a mapping sequencer that relies heavily on a reference genome, and a de-novo sequencer that constructs the genome from scratch. We concluded that if we were to use mapping algorithms we would lose valuable information that is critical to our pursuit, the reason is because our goal is to preserve mutations and irregularities throughout the genome, so we can analyse them thoroughly, however, once the mapping algorithm will attempt to compare between such irregular reads and the reference, it is highly plausible that it will result in dropping these reads because no apparent reference has been found. On the other hand, de-novo genome assembly is more likely to preserve the uniquety of the DNA, and as long as the **variance**, i.e the difference between reads originating from the same areas in the genome, is not exceptionally high we expect the assembler to keep the mutations.

The chosen assembler is **Minia** [6], a software for ultra-low memory DNA sequence assembly, that takes as input a set of reads and its output is a set of **contigs**, the result of putting together several shorter overlapping reads into a longer sequence, forming an approximation of a partial genome. Minia is based on a succinct representation of the de-Bruijn graph. The computational resources required to run Minia are significantly lower than that of other assemblers, yet it is only a contig assembler, therefore, it performs the first stage of a complete assembly in a computationally efficient manner but an additional software is required to assemble these short contigs into longer ones. In addition, Minia presents certain advantageous functions and tuning abilities, for example, changing the length of the k-mer to influence the length of the contigs, applying the parameter 'abundance-min' that helps in disposing of inherently flawed k-mer sequences that were sampled incorrectly, thus maintaining the balance between mutated reads and incorrectly dissected reads, and finally, Minia applies, by default, three methods of graph simplification, to assist in finding an Eulerian cycle efficiently: tip removal, bulges removal, and erroneous connections removal[7].

3. DICTIONARY:

   The next obstacle to arise is locating corresponding contigs between tumour and non-tumour tissues. A naive solution, though evidently inefficient, would be to compare every normal contig against every tumour contig. In our solution, we suggest the use of a dictionary dataset as a way to significantly reduce the number of comparisons, by grouping resembling contigs under the same hash bucket.

   The dictionary is an extensive dataset, constructed from the normal tissue, after Minia is complete. Each key is a k-mer taken from a 'sliding-window' over the contigs and its respective value is a list (also called **bucket**) of pointers to all contigs that contain said k-mer, such that, with every addition of a value, the list is scanned for replicas, to assure that each contig in a bucket is indeed unique.

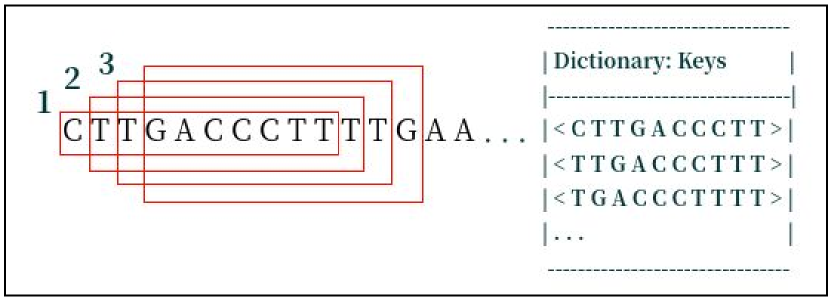

Fig. 6: Illustration of a sliding-window of size 10

The process can be explained as follows:

Parameter:	**Meaning**:
n	number of contigs for each tissue
X	average length of contigs
W	length of sliding-window (bases)

TABLE 1: Parameters for dictionary parsing

Assuming there are **n** contigs for each tissue and its average length is **X**. The naive solution for comparing each contig in tumour tissue to each contig in normal tissue would result in $\mathbf{O(n^2X)}$ operations. The maximum number of appearances contigs from normal tissue can achieve in a dictionary is X, that is because normal tissue contigs are parsed in an overlapping manner, with each step the sliding-window moves one-character forward. The maximum number of appearances that contigs from tumour tissue can achieve in a dictionary is X/W, that is because these contigs are parsed in a non-overlapping manner.

We propose setting $W = 10$, as a way to identify resembling parts and to overcome the Illumina

DNA sampling errors as well as the mutated bases. Illumina error-rate is estimated at 1% and our estimation to the error-rate of existing mutations in the genome is approximately 9%. In accordance to the pigeonhole principle, by parsing contigs into 10-character windows, if one out of 10 bases is a character that has changed, one of the windows will surely be errorless and free of mutations.
The probability of having a random window over {A, C, G, T} to equal a given window (i.e the probability of two different tissue-contigs to fall in the same hash-bucket) is: $\frac{1}{4^W}$.
Hence, for our solution, the probability for comparison is: $X \cdot \frac{X}{W} \cdot \frac{1}{4^W}$. For n contigs in the tumour tissue we compare X/W times with every contig in the normal tissue that is stored in the same hash-bucket: $n^2 \cdot X \cdot \frac{X}{W} \cdot \frac{1}{4^W}$
Once we look at the factor $\frac{X}{W} \cdot \frac{1}{4^W}$ and set W = 10, as we suggested, it would become: $\frac{X}{10485760}$
If we set X = 300 (average length of contigs): $\frac{300}{10485760} = 0.00002$
It is apparent that our suggestion reduces the number of comparisons of the naive solution by a factor of 0.00002 resulting in: $n^2 \cdot X \cdot 0.00002$

The dictionary is complete once all contigs of the normal tissue are stored accordingly. Thereafter, the algorithm inspects every contig of the tumour tissue, by comparing consecutive, non-overlapping windows (subsequences) of the contig, to the keys in the dictionary. Once a match has been found, we assume that the contig of the tumour tissue is likely to have its match in the bucket under said key.

Having discovered a potential bucket, we take each value within and along with the tumour contig and pass the pairs to an external method that locates and returns the overlapping subsequence between the two contigs. The reason why we are interested only in the overlapping sections between the normal and tumour contigs is because we expect to find the relevant mutations there, where the two contigs are relatively similar, but not identical.

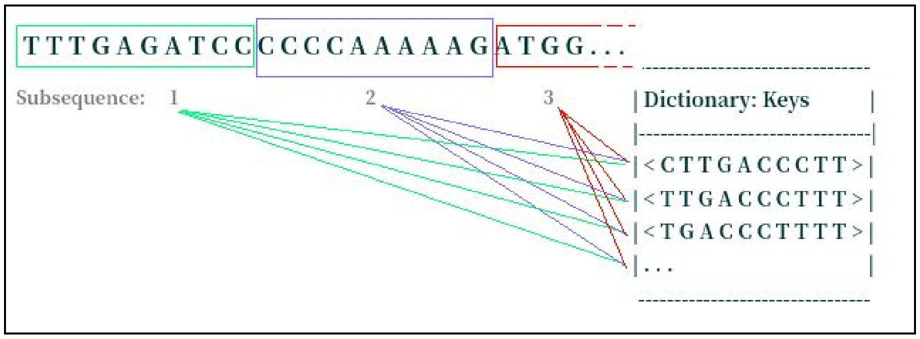

Fig. 7: Consecutive, non-overlapping subsequences of tumour tissue compared against dictionary keys

4. EDIT DISTANCE:
   Edit distance, a method of quantifying how dissimilar two strings are to one another by counting the minimum number of operations required to transform one string into the other, for example, to transform the string "kitten" into "sitting", three operations are needed ('k'→'s', 'e'→'i', ''→'g'). Due to the nature of our research, the implementation of edit distance quantifies the number of added, deleted and replaced nucleotides, all point mutation type variations, between normal and tumour tissue, of the same length. With the help of an open source code[8], we adjusted additional parameters to assist in quantifying our exact areas of interest. As of now, our research is still in an early stage and thus, does not provide advanced implementations for other types of mutations, nevertheless, we believe that the foundations we lay, can be used as a stepping stone to extend our research.

5. REPORT:
   The final phase is to sum up the findings of the analysis and formulate an informative mutation report, accordingly. The output of the algorithm is the results sent over from edit distance:
   - the quantity of each type of mutation (addition, deletion, replacement) and matches, overall.
   - The exact nature of the mutation e.g. if a replacement was detected, it will be noted what character was written and what character should have been instead.
   - The average length of contigs.
   - Sampling report : 1/100 from the total number of contigs, chosen randomly from edit distance, presenting how to turn one string into the other.

RESULTS:

1. MINIA:
   Our assumption, that applying the reads to the assembler would improve the duration of data analysis at a later term, needed to be tested, to reaffirm our advance. We designed a test comparing between the number of reads each tissue possesses and the number of contigs, the assembler outputs, for each tissue, while occasionally tweaking the parameters of the assembler to improve output. One parameter that was tested is **k-mer_size**, as we concluded that its value strongly depends on the input dataset. A sufficient value for short Illumina reads (>50) is k = 27, as for longer Illumina reads (≈ 100 bp) with sufficient coverage (>40), we settled with k = 43.

Data:

Number of reads	701,534,668
Min-length of sequence	100
Max-length of sequence	100

TABLE 2: Overview of the data

Minia Test Output:

k-mer_size	**31**	**31**	**24**
Special operations	none	no-bulge-removal, no-tip-removal	none
Number of contigs	54,175,480	27,250,890	35,424,317
Min-length of sequence	31	31	24
Max-length of sequence	17,348	29,033	8,215
Avg-length of sequence	78	114	81

TABLE 3: Review of experimental phase in Minia, according to changing parameters

From the displayed result we concluded to set k-mer_size=24 and use no simplification methods (bulge-removal, tip-removal), as these setting showed a plausible amount of contigs with an overall length that would allow us to properly utilise dictionary type dataset later on.

Length of contigs (bp)	**Amount of contigs**
0-100	1,416,863
100-200	751,294
200-300	502,735
300-400	360,979
400-500	874,569
500-1,000	357,765

TABLE 4: Quantities of contigs per length

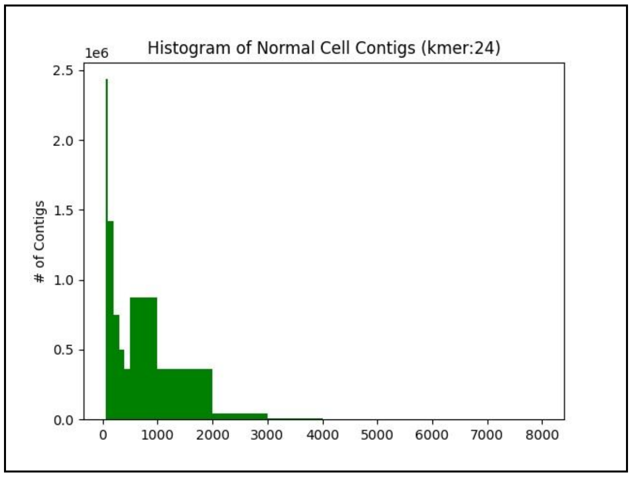

Fig. 8: Contigs length (no graph simplification): kmer=24

From fig. 8, it is evident that the majority of the contigs are between 0-1,000 bp long, hence we concentrated our efforts in these bounds to preserve unanimity.

2. DICTIONARY:
In the parsing of contigs into the dictionary we resorted to a limited amount of contigs than there are in the file, for the purpose of producing a theoretical evaluation for the amount of comparisons we may perform. An observation of the amount of contigs that are stored under the same key:

Number of contigs	1,000	100,000	1,000,000
Duration of parsing (seconds)	22.41	791.87 (~13 m)	58,926.88 (~16 h)
Maximum contigs in a bucket	482	8,739	67,371
Average number of contigs in a bucket	3.27	22.86	220.97

TABLE 5: Review of duration, length of contigs and the capacity of buckets

Therefore, we deduce that for a file containing 4,500,000 contigs:
- The duration of building a complete genome-dictionary is approximately 72 hours.
- The average number of contigs in a bucket is 995.
- The expected number of comparisons to all tumour tissue contigs is: $\frac{300}{10} \cdot 995$, where 10 is the size of the window and 300 is a number chosen randomly to be excessively higher than the average length of the contigs, to depict worst case scenarios.
- The expected number of comparisons overall is: $4,500,000 \cdot \frac{300}{10} \cdot 995$. the number of comparisons in a naive solution is: $4,500,000 \cdot 4,500,000$.
  The differences between the two solutions can be approximated as follows:
  $\frac{4{,}500{,}000 \cdot \frac{300}{10} \cdot 995}{4{,}500{,}000 \cdot 4{,}500{,}000} = \frac{199}{30{,}000} = 0.00663$

3. EDIT DISTANCE:
All tumour tissue contigs are compared by edit distance to normal contigs and the operations that are needed to transform a normal contig into the tumour contig are noted. Following the resulting distance, we observe the extent of the changes needed. We assume that whether the distance is bigger than 15% the size of the sequence, it is highly unlikely these contigs originated from the same area, hence they will be excluded from the final report. Nevertheless, the percentage was calculated approximately and can be reevaluated differently, depending on the amount of expected mutations in a tissue.

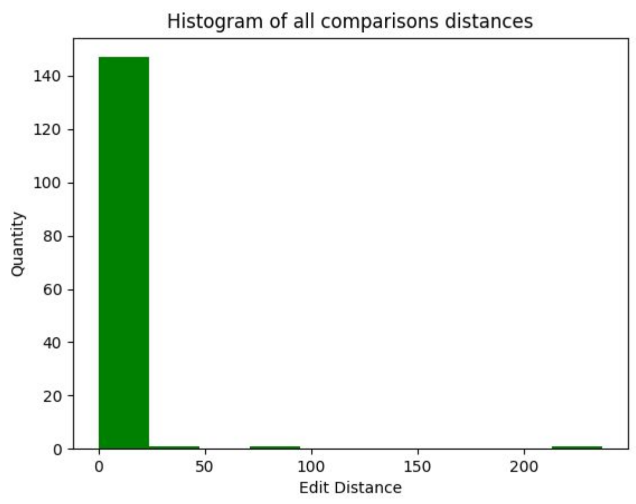

Fig. 9: Function of the number of contigs per Edit-distance between 0-250

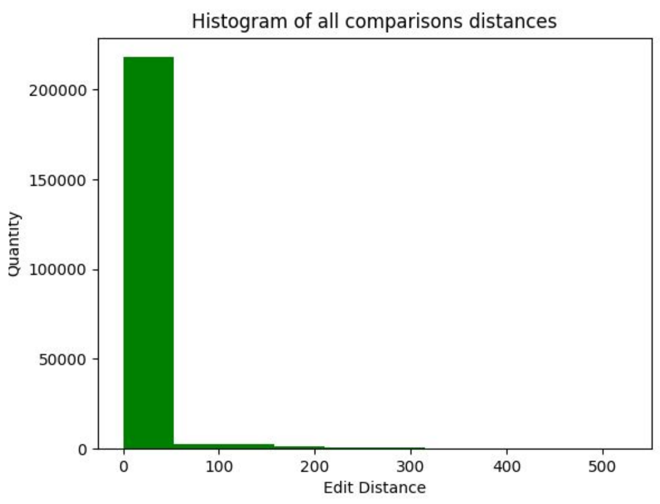

Fig. 10: Function of the number of contigs per edit-distance between 0-500

4. OUTPUT:

The algorithm outputs two reports:

1. An object of **mutation_distance** containing quantities of point-mutations divided according to their types and nucleotides, in addition to the percentage of mutation per strings' length.
2. Sampling file that is a collection of already compared strings and the distance between them to illustrate mutations that were found in the genome.

Mutation type	**Amount**	**Mutations per characters (%)**
insertion	5,769	0.19%
replacement	141,557	4.89%
deletion	5,769	0.19%

TABLE 6: Quantities and percentage divided between types of mutations

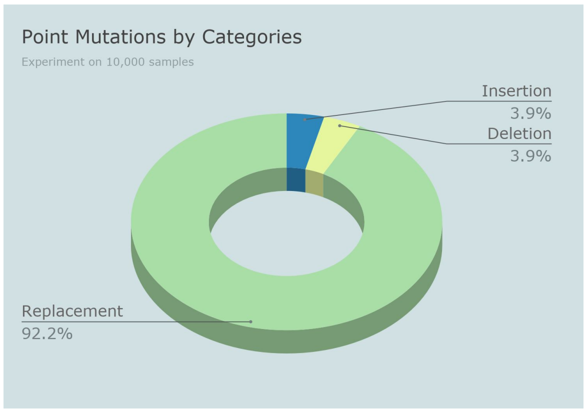

Fig. 11: Point Mutations by Categories

	'A'	'C'	'G'	'T'
Insertion	1,543	1,411	1,562	1,253
Deletion	1,829	1,444	1,184	1,312

TABLE 7: Quantities of insertion and deletion mutations divided by bases

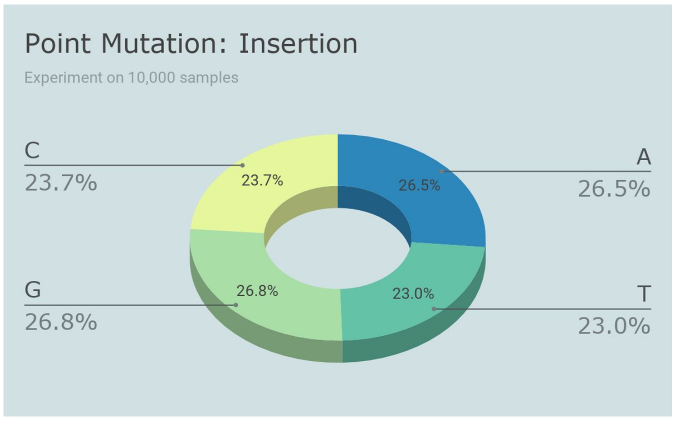

Fig. 12: Percentage of Insertion Point Mutation

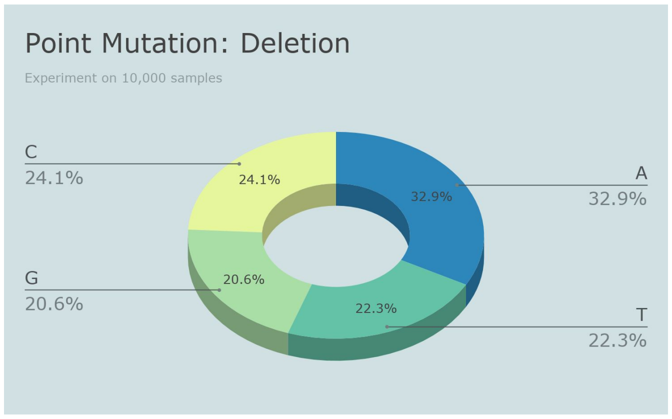

Fig. 13: Percentage of Deletion Point Mutation

	A→C	A→G	A→T	C→A	C→G	C→T	G→A	G→C	G→T	T→A	T→C	T→G
Replacement	9,359	18,605	6,106	9,221	9,533	17,711	22,062	8,907	7,164	6,665	19,678	6,547

TABLE 8: Quantities of replacement mutations divided by the changed bases

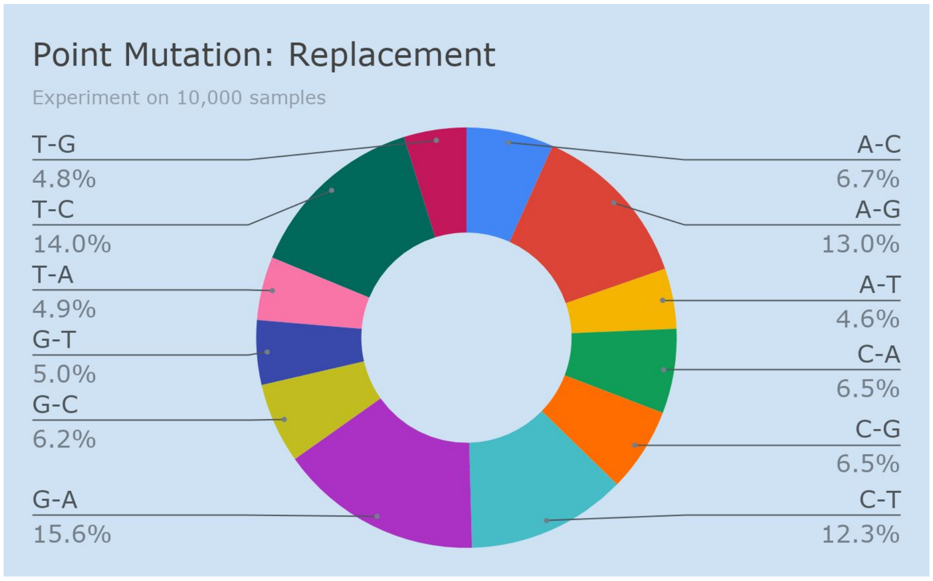

Fig. 14: Percentage of Replacement Point Mutation Types

CONCLUSION:

In conclusion, we believe that our research lays numerous primary methods that can be further utilised to broaden the borders of our algorithm. It is crucial to revise and reform our data, calculations, assumptions and conclusions to better detection, analysis and treatment provided by healthcare systems nowadays.
We suggest possible improvements that can be applied to the algorithm, such as: make use of the pair-end factor to generate contigs of higher quality, experiment with the parameters of the assembler, develop an applicable strategy of comparisons between longer string and for locating other mutation categories, and lastly, map mutations with regard to their location in the genome.

Our workflow was interrupted multiple times, due to limited resources and Amazon's computers crashing, therefore we were not able to complete more extensive tests on a larger amount of data, however, we are greatly thankful for this opportunity, to contribute, even mildly, to such an important field.

REFERENCES:

1. Jin, J. *et al.* (2019). "Identification of Genetic Mutations in Cancer: Challenge and Opportunity in the New Era of Targeted Therapy". *Front. Oncol.* **9**, 263 .
2. Heng Li, N. H. (2010). "A survey of sequence alignment algorithms for next-generation sequencing". *Brief. Bioinform.* **11**, 473.
3. Xingyu Liao, Min Li, You Zou, Fang-Xiang Wu, Yi-Pan, Jianxin Wang. (2019). "Current challenges and solutions of de-novo assembly". *Quantitative Biology*, 7(2): 90–109.
4. *Genetics Home Reference*. "What kinds of gene mutations are possible?" https://ghr.nlm.nih.gov/primer/mutationsanddisorders/possiblemutations.
5. *Illumina*. "Paired-End vs. Single-Read Sequencing". https://www.illumina.com/science/technology/next-generation-sequencing/plan-experiments/paired-end-vs-single-read.html.
6. Erwan Drezen, Guillaume Rizk, Rayan Chikhi, Charles Deltel, Claire Lemaitre, Pierre Peterlongo, Dominique Lavenier. (2014). "GATB: Genome Assembly & Analysis Tool Box". *Bioinformatics*, 30(20): 2959–2961. https://doi.org/10.1093/bioinformatics/btu406.
7. R. Chikhi & G. Rizk & E. Drezen & D. Lavenier. (2018). "Minia-GATB— Short manual". https://github.com/GATB/minia/raw/master/doc/manual.pdf.
8. Ben Lambert. (2018). "edit-distance". https://github.com/belambert/edit-distance.