IDENTIFYING AND CATEGORISING MUTATIONS IN TUMOUR CELLS AS A MEANS TO ADVISE TREATMENT

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

2020

Personal Notes:

The structure of this paper will be as follows:

- **Title**
- **Abstract: 1 paragraph (<250 words)**
- **Keywords**
- **Introduction: 1.5-2 pages**
- **Methods: 2-3 pages**
- **Results: 6-8 pages**
- **Discussion: (<6) pages**
- **Conclusion: 1 paragraph**
- **Figures, Tables**
- **References**

Abstract

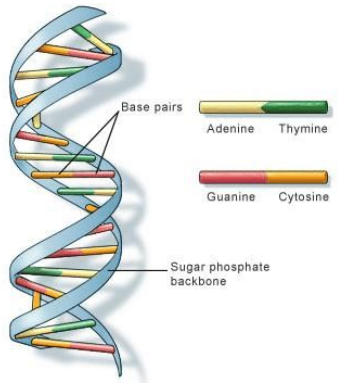
In this paper, we will review and present the results of our research and development for constructing a feasibly running algorithm that once given a collection of unorganised reads (strings of nucleobases A, C, G, T), and when assembled correctly, they construct the full specimen genome; the algorithm uses them to map the genome of the specimen using overlaps between the reads, with the help of an external existing program. Once the genome is fully mapped, the algorithm searches for mutations of certain types which, in turn, when evaluating the number of mutations, categorising mutations and identifying precise changes in comparison to a normal genome, can lead to better detection, diagnosis and treatment provided healthcare.

Keywords:

INTRODUCTION

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. 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 (A, T, G and C) and which results with a collection of unorganised, partially overlapping reads, that are fragments from a longer DNA sequence.

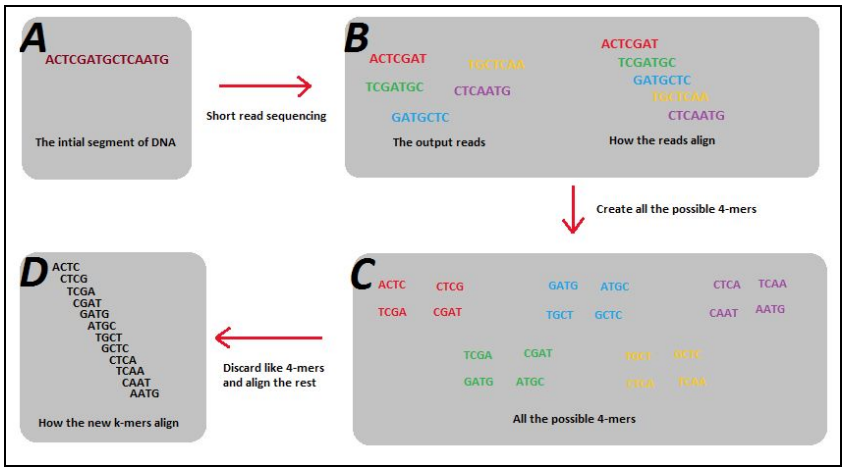


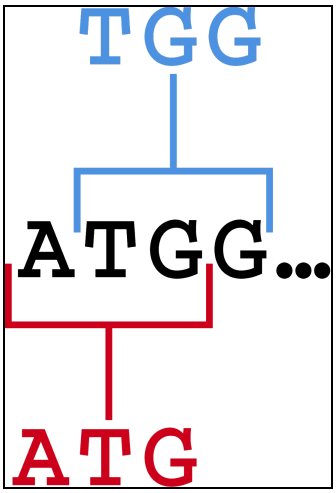
Every such collection has a **coverage** (also, **cover number**), which is the average number of reads that sample a single nucleotide, and can be calculated as such:

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

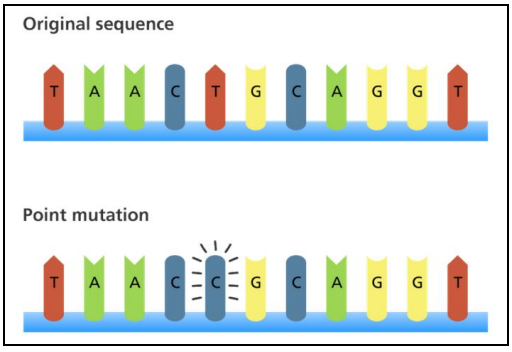
These reads are aligned and merged, using their overlapping and a variety of unique algorithms, in order to reconstruct the original sequence, in a vast and intricate process referred to as **sequence assembly**. This process contains many different approaches and algorithms and in the following paragraph we review the ones most relevant to our research.

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, that is in fact a guideline reference to the genome of the specie, resulting in an assembled genome that is similar but not necessarily identical to the , and **de-novo** [3],which consists of assembling short overlapping reads to create full-length sequences. 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.





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.



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 of full genome assembly and mutations 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 healthy tissue and the other is a tumor cell. 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 average cover number of 100-base reads. The inputs are in FASTQ file format named *sample_number-T_R1* for the tumor tissue and *sample_number-N_R1* for the healthy tissue, 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; however we negated this approach for 2 reasons: firstly, while taking into account that there are over 700,000,000 reads, that will inevitably strain 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, that once we compare between all reads, would result in an inaccurate statistic. Lastly, we intend to enable future advancement of inferring mutations to specific areas in the genome, through the connections we create between the reads and their placement.

   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, that 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 assembler we used 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 the expected 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, using 'minia-pipeline' on top of Minia to run the assembler consecutively in search of the ideal k-mer that results in longer contigs, applying the parameter 'abundance-min' that helps in disposing of inherently flawed k-mer sequences that were read wrong, initially, 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]. We selectively asked to not perform parts of those operation through those command line arguments: -no-tip-removal, -no-bulge-removal

3. DICTIONARY:

The dictionary is an extensive dataset, constructed from the normal tissue, after Minia is done. Each key is a k-mer taken from a sliding-window over the contigs and its respective value is a list 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 value list is indeed unique. The length of the k-mer derives from the percentage of errors found in a read through a calculation that can be explained as follows:

Parameter:	**Meaning**:
N	number of contigs for each tissue
X	average length of contigs
W	length of sliding window (subsequence)

A naive solution to locating corresponding reads between tumour tissue and normal tissue would be to compare every normal contig in the dictionary against every tumour contig, though it is clearly inefficient. We suggest setting W = 10 in addition to implementing a hash-function to store contigs' pointer in the dictionary. The maximum number of appearances that contigs from normal tissue can achieve in a dictionary is X, that is because 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 tumor tissue can achieve in a dictionary is X/W, that is because these contigs are parsed in a non-overlapping manner.

The probability of having a random window over {A, C, G, T} to equal a given window is: $\frac{1}{4^W}$, hence, for our solution, the probability is: $X * \frac{X}{W} * \frac{1}{4^W}$, i.e. for n contigs in the tumor tissue we compare X/W times with every contig in the normal tissue that is stored in the same list value: $n^2 * X * \frac{X}{W} * \frac{1}{4^W}$

Once we look at the factor $\frac{X}{W} * \frac{1}{4^W}$ and set W = 10, as we suggested: $\frac{X}{10485760}$
If we set X = 100 (average length of contigs): $\frac{100}{10485760} = 0.00000953674$
It is apparent that our suggestion reduces the number of comparisons of the naive solution by a factor of 0.00000953674 resulting: $n^2 * X * 0.00000953674$

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 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 under said key.

4. EDIT DISTANCE:
   At this point, the algorithm goes over the values of the key, while utilising **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 tumor tissue. 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.

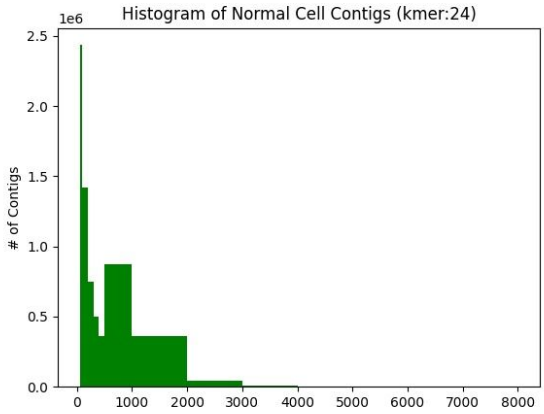
Data:

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

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

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.



Contigs length (no graph simplification): kmer=24

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

Looking at the histogram, 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

Therefore, we deduce that for a file containing 4,500,000 contigs:

- The duration until completion 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: $\frac{300}{10} * 995$, 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: $4,500,000 * \frac{300}{10} * 995$. the number of comparisons in a naive solution is: $4,500,000 * 4,500,000$.

  The differences between the two solutions can be approximated as follows:
  $\frac{4,500,000 * \frac{300}{10} * 995}{4,500,000 * 4,500,000} = \frac{199}{30,000} = 0.00663$

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.

DISCUSSION:

Although we had these paired reads, we decided to ignore the pairing information. As part of the improvements for advanced research, merge can be done to more reliable input data. (INPUT REVIEW)

CONCLUSION:

REFERENCES:

1. Jin, J. *et al.* Identification of Genetic Mutations in Cancer: Challenge and Opportunity in the New Era of Targeted Therapy. *Front. Oncol.* **9**, 263 (2019).
2. Heng Li, N. H. A survey of sequence alignment algorithms for next-generation sequencing. *Brief. Bioinform.* **11**, 473 (2010).
3. [No title]. https://link.springer.com/content/pdf/10.1007%2Fs40484-019-0166-9.pdf.
4. Genetics Home Reference. What kinds of gene mutations are possible? https://ghr.nlm.nih.gov/primer/mutationsanddisorders/possiblemutations.
5. Paired-End vs. Single-Read Sequencing. https://www.illumina.com/science/technology/next-generation-sequencing/plan-experiments/paired-end-vs-single-read.html.
6. GATB. GATB/minia. https://github.com/GATB/minia.
7. [No title]. https://github.com/GATB/minia/raw/master/doc/manual.pdf.
8. belambert. belambert/edit-distance. https://github.com/belambert/edit-distance.