National College of Ireland

Pattern detection in genomic data for mutation in nucleotide sequences

MSc Research Project
Cloud Computing

Akshay Tak
x16110234

School of Computing
National College of Ireland

Supervisor: Victor Del Rosal

National College of Ireland
Project Submission Sheet – 2015/2016
School of Computing

Student Name:	Akshay Tak
Student ID:	x16110234
Programme:	Cloud Computing
Year:	2016
Module:	MSc Research Project
Lecturer:	Victor Del Rosal
Submission Due Date:	16/08/2017
Project Title:	Pattern detection in genomic data for mutation in nucleotide sequences
Word Count:	5269

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are encouraged to use the Harvard Referencing Standard supplied by the Library. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action. Students may be required to undergo a viva (oral examination) if there is suspicion about the validity of their submitted work.

Signature:	
Date:	15th September 2017

PLEASE READ THE FOLLOWING INSTRUCTIONS:

1. Please attach a completed copy of this sheet to each project (including multiple copies).
2. **You must ensure that you retain a HARD COPY of ALL projects**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. Please do not bind projects or place in covers unless specifically requested.
3. Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Pattern detection in genomic data for mutation in nucleotide sequences

Akshay Tak
x16110234
MSc Research Project in Cloud Computing

15th September 2017

Abstract

As the implementation of sequencing technologies increases, the data generated using the traditional NGS methods gets more and more high definition. This restricts the processing capabilities to a certain extent specially in terms of time. Another important aspect that comes into picture is the propensity for the usage of this data. This research aims at working on both these platforms. The idea is to use an approach to analyze a small part of the data in the genome and use it to detect genetic similarities in the form of A,G,T,C patterns. Along with this pattern matching by parsing the data, this research will also focus on the time taken for these processes as that is another big concern of the genomics big data. All this will be executed and the results based on small segments of the data in a big data environment using big data processing resources have been made to ensure that this research is ready to migrate to the batch processing in cloud with future aspects of real time data in the same (which is beyond the scope of the time frame of this research).

1 Introduction

Next generation sequencing has been around for more than 30 years and its contribution to understanding the human genetics. With the amount of data being generated over the past few years with further advancements in technology, the data generated is in high resolution and has hence scaled vastly. This means that the ability to analyze this has to scale as well.

In the world of modern genomics, an important aspect for any kinds of analysis is the processing of data which is generated. One of the biggest problems with the data generated in this field is that it is exponentially incremental and the processing parameters are limited. To make sure that the data generated is of use and not just taking up storage, we have to assure that its processing is done in a proper manner. This is what motivated this research. The original data generated by FastA processes is not capable of proper processing due to many factors such as the scalability of data which exceeds the processing capabilities of many schedulers. Furthermore, to add on to this problem, even if the data is processed somehow, analysis is again a big issue. This means that using this data requires a whole lot of effort.

To ease these problems, this paper presents a methodological approach towards comparing the processing times within the genomic data sets. The idea is to use DNA data sets which are scalable for size in text files for up to 3GB and beyond, along with RNA data sets which are smaller in size but must be analysed in groups. The idea is to check how to make these datasets capable for analysis and work on analysing patterns in the same. The impact of this research is that once the patterns can be recognized in the genomic data, we can store the patterns of mutated genomes. This means that if out of a set of 100 samples there are 10 people with same genetics with similar diseases, their genetic code can now be used as a sample to detect early symptoms of the disease from the samples of other people who could possibly display similar symptoms.

To do so, this paper has been divided into different sections, each of which contributes to the different aspects of this research. The first section is the literature review, which throws light on the basics of this research and how Next Generation Sequencing works in general, along with a highlighted layout on how everything is processed in state of the art methods such as the sequencers which are used. The next section defines the methodology followed for this research which gives a brief account on how this research was executed and the pattern of research flow of this research. The next section has been dedicated to an elaborated explanation on how this research and its outcomes have been implemented. This section decides whether the research was successful in determining the processing capabilities of the whole genome and whether pattern detection is possible in the genomic database or not. The outputs of this section are discussed in details in the end where all the algorithms have been implemented and methods have been used to not only measure the time taken for processing in the DNA and the RNA databases but also show their pattern mapping capabilities using cloud environment.

This approach has been coined as a probabilistic one because due to the limitations of this research in terms of time frame, a small segment of the genome is analysed and processed for time and pattern detection instead of the whole thing, and the rest is considered to be falling into place in a probabilistic manner, any further problems which could be faced on scaling this research are in future works section. To achieve this, the underlying programing language used is Java and cloud computing tools have been used. Hadoop architecture has been used to provide the environment for this research. To enable real time data read, tools and extensions have been used which are discussed in more details in the implementation section. The final evaluations have time based recordings of everything done and pattern mapping in the FastA generated code.

2 Related work:

2.1 Next Generation Sequencing

In 1977, Fred Sanger and Alan Coulson proposed the idea of using genetic deciphering for genes and then the entire genomes, this process as we know today, led to unparalleled research and advancement in biological studies. Next Generation Sequencing has been around for over 39 years, having been one of the leading factors in the enhancement of forensic sciences applications, its now time that the information that has already been generated be used in a more productive manner. This idea led to the research of genomic patterns in different regions of the world and for different types of diseases. This was

a corollary to the thoughts of Yaffe and Tanay (2011), who predicted that every kind of genetic mutation can be detected using a probability based approach. This idea as a whole, was not just revolutionary, but also controversial in many aspects. Though the idea of tracing genetic patterns in similar symptom displaying victims sounded like a concrete approach, the vagueness of this methodology has always been criticized. This was primarily due to two factors, first, diseases which were hereditary could be traced but diseases which are not cannot be. The second factor is that until 2009, high definition data was not generated, which meant that the information to be parsed was in a low resolution, which was definitely a barrier in terms of research.

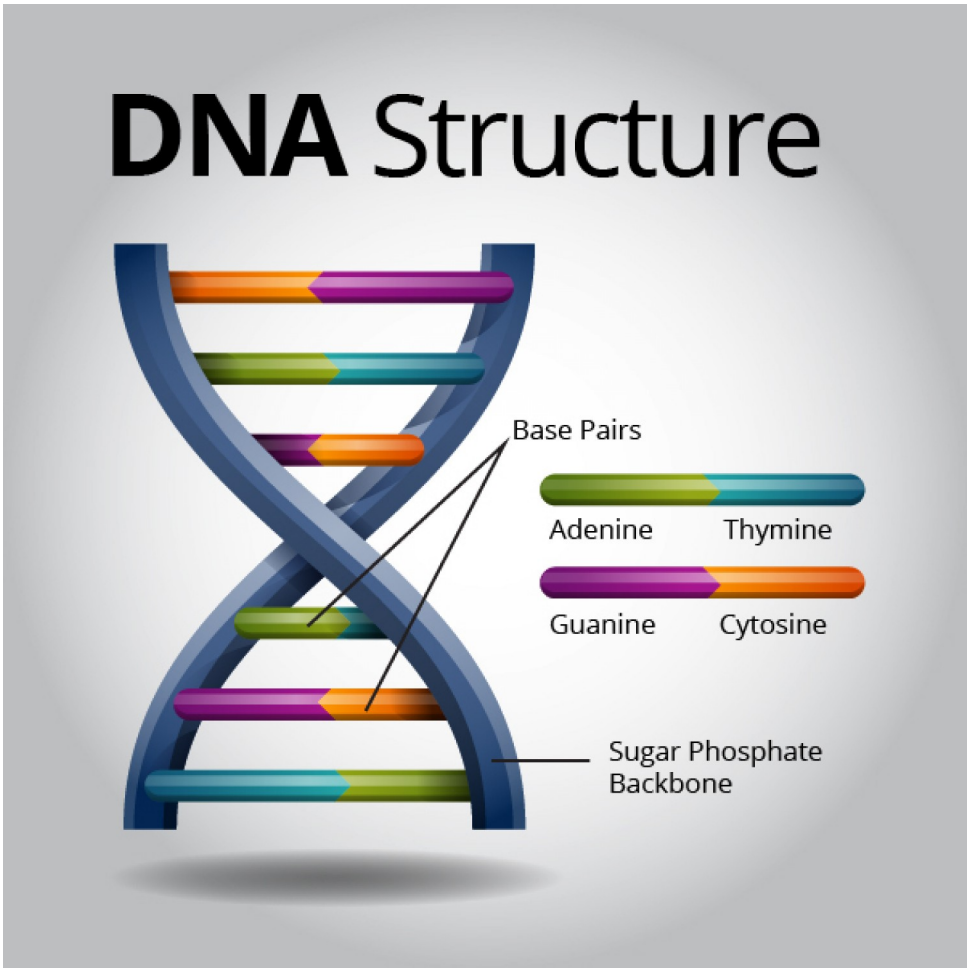

Figure 1: DNA Structure

The current methodology for this process works as follows, in a sequential manner. The genomic data is first transferred in a raw form and heads towards the processing units for fragmentation. Here, the data is prepared for any kinds of MapReduce McKenna et al. (2010) operations on it. This process outputs a linear format of the data, this format is the one that comprises of elements of the DNA prepared for the sequencing process and which are linear. For any kinds of comparisons, this data is then set through the DNA library. For the purpose of this research, the DNA library that has been incorporated will be a pre-existing one with AWS where they provide Data as a Service and there is a referral genomic data.

This data is now prepared for the testing purposes for checking how the sequencing works. For that, the data goes through a PCR process. A PCR process (polymerase chain reaction) is where the DNA sample is treated against different enzymes which behave neutrally towards all the nucleotides but one (Livak and Schmittgen; 2001). This means that to detect the nature and presence of the 4 primary nucleotides, we use 4 different enzymes which are each characteristic to activate any one nucleotide, i.e., acetone, guanine, cytosine or thymine. One of the things which are considered during this process is also the fact that in some cases, these enzymes also activate the hybrid nucleotides. One of the examples of this is the DNase enzyme, which is responsible for activating Cytosine in such processes, which also is a key factor in the activation of Purine and Keto, which are both hybrids with traces of cytosine and major contributors being acetone and guanine respectively. This has led to a slight error but due to high resolution outputs, we are now able to distinguish between them and avoid any such errors in any research. This research in particular, implements the usage of high resolution outputs to obtain more precise data for pattern recognition and matching.

Once this process is taken care off, next up is the storage part, this is done on the cloud. The instances for which will be deployed on OpenStack. Once the storage is done, the real processing of the data begins. This data is already passed through the enzyme reaction stage and hence, the outputs are based on the respective nucleotides which react to the different enzymes (Livak and Schmittgen; 2001). The obtained output is generated using Hi-C. The reason for this is that Hi-C is used to generate data strings of up to 6 base pairs, which is double as compared to the output of 3C. This has been explained in detail in the next section of this report. Next up is the initiation of the analysis part of this output. For this, we will use the tool HiCup (Wingett et al.; 2015), this tool is one of the most popular tools for the analytics of Hi-C outputs. It performs statistical mapping and filtering on the data which is generated from the Hi-C. This tool uses the programing language HOMER Seitan et al. (2013), which has been prepared with the sole purpose of supporting Hi-C analytics. It comes with several programs which are going to be implemented and patterns which will be followed during the process of genomic pattern recognition. A corollary of this methodology and its implementation is also under testing during this research where this prepared algorithmic approach will be tested for machine learning in pattern recognition as well, when there will be an implementation for how the various machine learning algorithms can adapt to the incoming patterns for future reference and how they will be stored in a library with high processing rat using a suitable GPU.

Once the HiCup process is done and the data is analyzed for any problems such as integrity and missing features, it is prepared for the output. Any output in the genomic process is done using FastA or FastQ. These 2 convert the HiCup data into readable format in a text file. For this research, Langmead et al. (2009) FastA is used for its efficiency in high resolution data processing. This leads us to generating statistical data which is viewed in a tabular manner. This table comprised of 60 columns and approximately 55 million rows in entirety for a single genome.

2.2 Hi C

While normal genomic sequencers produce outputs of the lengths of 1 or 2 base pairs of nucleotides at once, Hi-C [2], a high-resolution enhancement of the 3C (chromosome conformation capture) model generated the output at up to 6 base pairs per sequence. This, despite being better than everything else being implemented, was still not good enough for high quality outputs. In all its glory, this method, if combined with FISH, was what helped high resolution reads for sequencing (Van Berkum et al.; 2010). As mentioned earlier, Hi-C uses Homer programs. Homer programs have often been disputed in terms of their integrity. This is due to the fact that their outputs can be rigged, which distorts any kind of belief in their output. There are some highly efficient programs in Hi-C, such as automated PCA analysis, making tag directory, etc. which are specific to their contribution towards analysis of any Hi-C outputs.

2.3 FISH

Fluorescence in Situ Hybridization (FISH) approach was first introduced to RNA nucleotide sequencing [2] and then found its way towards Genomic. This approach involves the usage of illuminating the target sequence parts. This allows us to separate the denatured DNA probes from the normal ones, hence eliminating the mutated parts of the DNA. These mutations in DNA can be treated as leading patterns to understand what to expect when we combine this research with the probabilistic model of Yaffe and Tanay. A combination of both the models would lead to a very precise approach towards the detection of genetic mutations.

FISH incorporates state of the art memory allocation to process the output of all the sequencing methods. Since this research focuses on the performance enhancement aspect of the genomic field, any kind of enhancement in the memory allocation and processing is a bonus to the overall process. One of the biggest things to be understood in NGS is that the research requires GPUs for processing. Since a GPU takes care of the processing capabilities of the process, we need a library which is compatible with the high performance metric of the GPUs. This is where high performance libraries such as Meusli and Skepu come into being. Both these libraries are opensource, and can be implemented in high performance paradigms such as genomic computing and are both operable and programable on C++, which makes them more user friendly and basic in implementation. Among these 2, Meusli has a better scope of implementation towards this research due to its internal MPI which allows the different libraries to communicate, which can further be easily linked to the Scala interface while implementing machine learning algorithms for pattern recognition in Scala. The base of all the programming, whether in the Scala interface or the pattern libraries is object oriented, which means that any kind of cross platform transactions which arise can easily be implemented. This is again a huge bonus while working with pre-programmed editable libraries such as Muller-Funk et al. (2009) Meusli, that any cross platform problems which are to be faced later are already prepared for.

2.4 Why this research?

The prime motivation behind this research is to understand and decipher the genomic pattern of cancer prone genetics. As stated in earlier sections, the approach needs a high-resolution sequencing output and a combination of FISH with Hi C would certainly do the trick. As this approach would remove any discrepancies in the data and give out an HD output. The data which is to be read will be generated using the Illumina sequencer. The reason for using Illumina over other sequencers is primarily its availability, lost cost and its high stability (Patel and Jain; 2012).

This research on successful implementation will lead the way towards preparing a better analytical approach towards pattern detection in genomic data, which is crucial for the detection of diseases in a genomic pool. This could definitely pave the way for advanced medical sciences and help humanity in fighting its way through various diseases which can be detected before-hand and cured. This will also help us understand how advanced genetics are slowly coming into picture and how it has changed the entire game. With a sharpened comparison between DNA and RNA genomics in homogeneous and real-time genetic inputs, this research will cover all paradigms of genomic analysis and output the best methods to establish a pattern recognition format.

2.5 State of the art methods and research

Current scenario in the genomic sciences applies very simple approach, the sequence of operation of the process is as follows. Genomic data is generated using sequencers such as Illumina and Rothberg and Leamon (2008) Roche 454, this data is interpreted and analyzed for any discrepancies which are then identified for patterns. These patterns, in the proposed research will be used as the base. The idea is to store the patterns and run a check for the same or similarly occurring patterns in the sequences of the samples to be tested. This research will be conducted in two main phases, the first phase will involve detecting the cancerous genetic code in a given sample and the second phase will involve using the information to precise down the pattern and implement its search on a given genomic dataset. The sequences taken under consideration would only be Genomic and will not constitute RNA nucleotide sequences. The nucleotides which will be looked after in patterns would be Acetone (A), Cytosine (C), Guanine (G) and Thymine (T). The reason for leaving out RNA nucleotide in the output research is that in RNA, the nucleotides pair up to form higher level protein which require even higher definition of data for processing, which is beyond the scope of this research.

2.6 Configuring the Hadoop Cluster:

The Hadoop cluster is needed to upload and analyze the data in HDFS. For this to execute comfortably, we need to determine certain elements of the cluster before we can prepare for any kinds of analysis or programing in the cluster (Zikopoulos et al.; 2011).

This part of the research involves using theoretical knowledge of the HDFS file system and using it to understand how it changes the data usage. In case of our RDBMS data

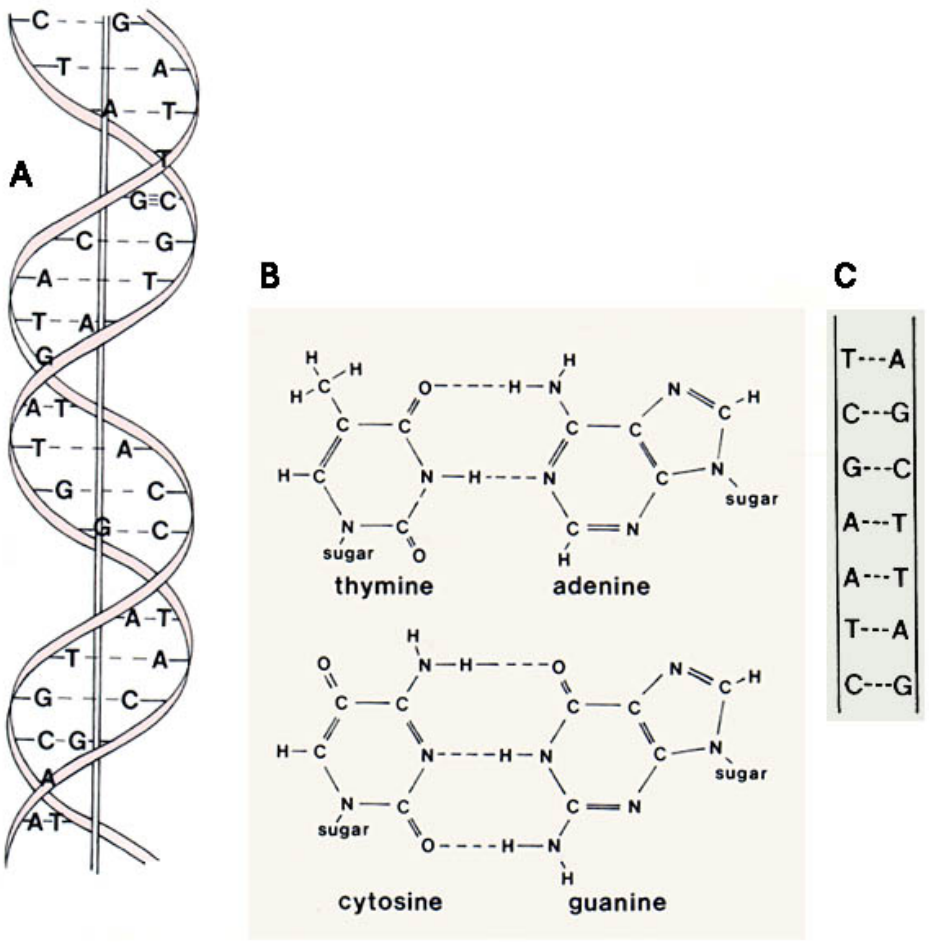

Figure 2: DNA Double Helix

from DNA, the file is 600 Mb in size, where as in RNA, the file is just 30 to 40 MB in size. The idea is to configure the Hadoop cluster in such a manner that when the blocks are being made for map reduce in batch process, their size is similar to what is expected in RNA. For a 660 Mb file, the number of blocks created to keep the size between 30 to 40 Mb is approximately 22. This will be the replication factor of the Hadoop cluster while doing the processing on the DNA genomic output. Further, the HDFS is used for storage in DNA genome only, for RNA, we will implementHan et al. (2011) Cassandra instead of the earlier proposed MongoDB.

There are multiple reasons for replacing MongoDB with Cassandra in the ending implementation stage. First of all, Cassandras performance in terms of massive data set processing has been widely acknowledged over MongoDB Li et al. (2009). It is flexible in case of high number of columns and considering that our outputs have 60 columns to be analyzed and linked, this seems like the right choice. MongoDB is the best choice in case the data was stored instead of real time connected to the server, as in this case.

3 Methodology

To implement this research, the process required high detailed outputs and considering the given time frame, the following assumptions will be made.

First, the data obtained is not restricted to the format depicted in the research. It comprises of various elements apart from the four nucleotides. The research focuses on establishing similarities amongst the patterns detected in the formation of nucleotides (A, C, G, T).

Another assumption made is that the genomic structure obtained from both DNA and RNA is subjectable to similar kind of analysis. The reason for doing that is that the research is to establish the efficiency of pattern mapping for future analysis for purposes such as cancer detection using just the genomic data.

The third and final assumption made is regarding the proximity of the type of data being used. While processing any data set of sizes as big as the ones here, there is a constant problem of how the data is uploaded to the Hadoop cluster, its storage methods and hence, the methods used for its analysis. Our data is both RDBMS and non-relational. The data which is loaded to the HDFS directly will be from the 1000 Genome project with fixed data batches and files. The genomic data to check the processing capabilities of the algorithm in both RDBMS and non-relational database will be generated from DNA and RNA respectively.

For the genomic data in DNA, the batch size of the files used will be approximately 600 Mb, and per DNA sample there is a single file generated. The execution path for this is as follows, first, the data will be uploaded to the Hadoop cluster and it will be stored in HDFS. As per the normal functionality of HDFS, the data will be stored in blocks. The configuration of the Hadoop cluster will be done to assure that all the blocks are of small size and the process involves a high level of parallel computing. The default replication factor of HDFS will be changed to 6, and the block size will be chosen to accommodate the file with a commendable maintained gap of 660 Mb to accommodate any mutations observed. This process once completed, would allow this research to focus on an interface to show the entire implementation. This interface will be provided by the HDFS browser and the code containing the pattern mapping features would be written in Java. This code will apply variables which can store patterns for different genomes in different codes as predefined methods for future reference. This will lead to this research resulting in a library of patterns to refer to in case it needs to match patterns. The idea is to use an enhancement model for the research which will prepare any future genomic data to be processed against the data of a genome with the desired characteristics. These characteristics can be mutations such as extremely athletic genetics and other enhanced features, as well as diseased genetics for proper and timely cure and prevention of diseases such as cancer cells leading to a specific genomic pattern.

The other type of data would be received from the RNA molecule of the body. For the purpose of comparison in this research, the RNA data will be run in a real time environment to make sure that all the outputs are processed in a manner where non relational outputs are also considered. This will allow us to maintain a record to see whether the algorithm for mutation and pattern detection worked better in batches or

real time. To facilitate real time processing, the idea is to use Cassandra for storage. The entire framework will be replicated in Spark cluster and the output will not only be stored in a library but also uploaded to a machine learning algorithm. This part of the research is not feasible in the time period but the ground work for future implementation in this direction will be established during the research work. The idea is to understand and implement the concept of RDDs in Spark ecosystem for real time processing. The problems in this area are that for comparison purposes, the data size should have been similar, i.e., 600 Mb. But, the size in RNA genomic output is approximately 30 to 40 Mb per file, which means that as the RDDs are being filled, calculations have to be made to make sure that the block size in batch processing in DNA genomic output is adjusted appropriately for more accuracy.

One of the leading factors in this research is the algorithm which is to be used for pattern detection. The idea is to implement this code in Java, with basis logic for any kinds of text input. This is possible due to the fact that any outputs which are generated in the genomic fields, irrespective of the data type being real-time or batch, have to go through FastA or FastQ, which means that the output file is going to be in txt format. This is one of the biggest advantages of using Genomic data for this kind of research to analyze the patterns with differently behaving data inputs.

4 Implementation

To successfully implement this research, one of the biggest issues faced was the conversion of the FastA generated txt file. The reason for this is that the file is available in a downloadable format, but the download doesnt complete and the extensions cause the file to link up to the next genomic file without completing the first one. The result of this is a forever downloading chain of files which have to be processed. To fix this, the first genomic file was interrupted and the data was taken. Next up was the usage of this data and storing it somewhere for processing. To do that, we have used the HDFS storage which is setup on Hadoop. The next problem faces were to parse the data. Now, the data which was generated was in a text format which was stored in arrays in the form of a single string per row. This data needed to be parsed and to do that, a Java code was written which can separate the values in a string and process them as a single input at a time. Though this process adds a constant time difference to the whole process, it is crucial for high definition outputs when it comes to pattern matching for any kinds of genetic analysis in mutation or disease detection from available patterns.

Now, to migrate the outputs which are all text based, MySql has been implemented in the backend. The implementation done is working in the following sequence of operations:

1. The data is set on download from the following link:
https://www.ncbi.nlm.nih.gov/nuccore/$XR_4$26948.3?$report = fasta$

2. Once the DNA FastA sequences are set for parsing, we interrupt the download at one genome.

3. To do this, a manual process is followed where the moment a limit of 1GB on the txt file size is exceeded, the download is paused and the data is now used for processing.

4. The next step is to store this data in HDFS.

5. Once this process is done, the data is now ready to be parsed, this is done using Java program.

6. The program can now detect any inputs in the FastA string as individual strings.

7. Now, comes the part where this research benefits the community, the pattern detection. For this, the codes have again been implemented in Java.

8. Another important aspect of this research is a comparison between the time taken by the DNA and the RNA based genome for processing and pattern matching. This is required due to the varying nature of the 2 kinds of data. The DNA behaving like real time and the RNA like batches.

9. For the purpose of this research, these problems have been fixed and the real time data has been prepared for large batch processing. This again adds a little time to the overall process but in the long run, it would benefit the research as we need to test the pattern making in this environment.

10. If successful, the methodology and implementation of this research can always be migrated to environments like Spark and Sqoop variables can be used to implement the big data and real time analysis can be performed based on this research as the foundation and these codes as the prototypes of the same.

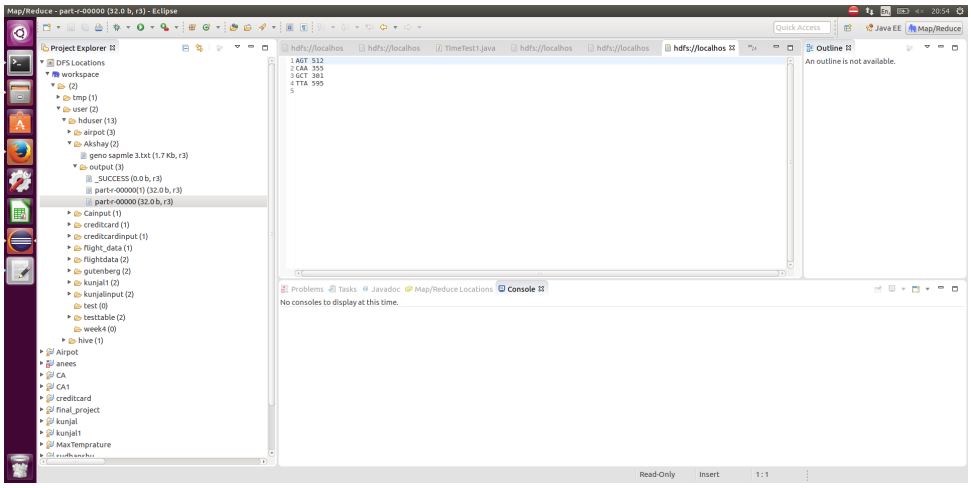

Figure 3: Running output

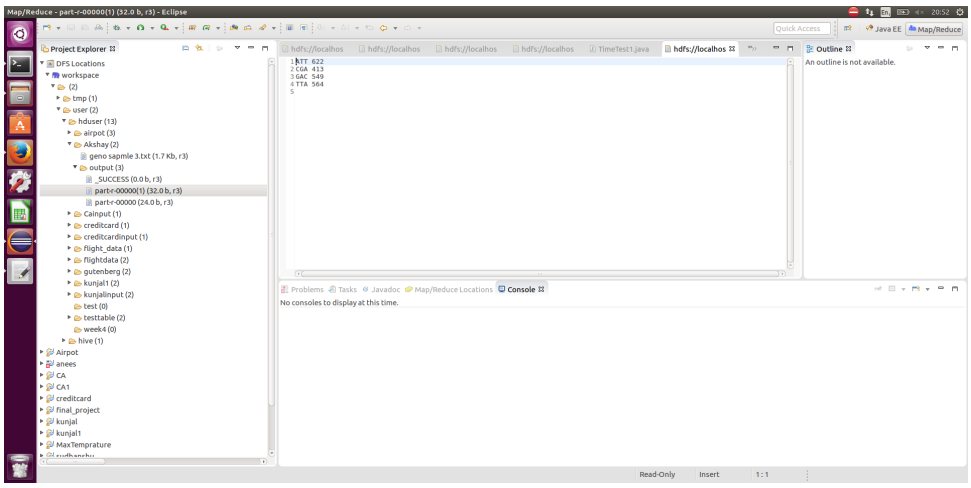

Figure 4: Running output

5 Evaluation

For the purpose of evaluating the researchs outputs, we have 2 parameters. First parameter is to compare the time taken to prepare the whole data for analysis in terms of both DNA and RNA genomic structure based on the FastA outputs. Another thing that has to be evaluated is the processing time which is taken to analyze the patterns which are to be detected in the output generated. The results of these evaluations have been shown below:

With existing patterns in FAST A output		
DNA String size 3	Time	
Run 1	12.55	
Run2	13.15	
Run 3	14.58	
Run 4	11.62	
DNA string size 4	Time	
Run 1	39.44	
Run 2	44.86	
Run 3	39.42	
Run 4	55.21	

Figure 5: With Exiting patterns in FAST A output

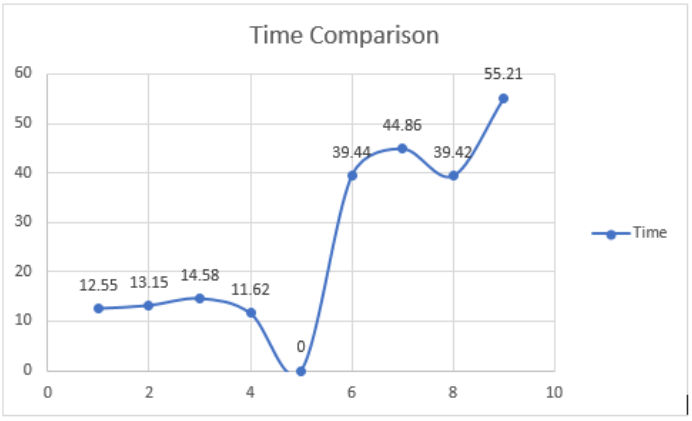

Figure 6: With Exiting patterns in FAST A output

With existing patterns in FAST A output	
RNA String size 3	Time
Run 1	4.2
Run2	2.5
Run 3	3.7
Run 4	5
RNA string size 4	Time
Run 1	1.4
Run 2	5.7
Run 3	6
Run 4	3.7

Figure 7: With Exiting patterns in FAST A output

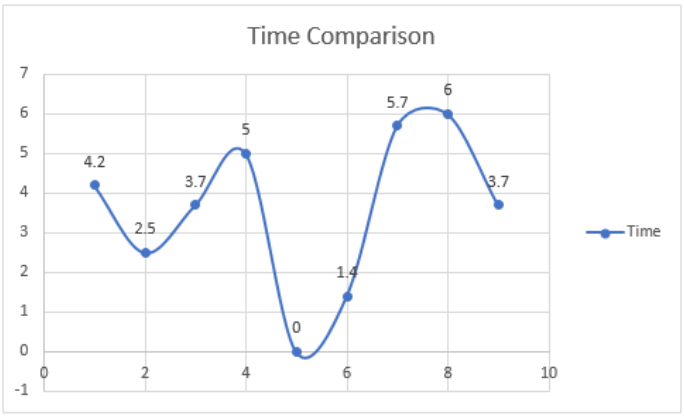

Figure 8: With Exiting patterns in FAST A output

Without existing patterns in FAST A output	
DNA String size 3	Time
Run 1	3.4
Run2	2.1
DNA string size 4	Time
Run 1	1.3
Run 2	1.1

Figure 9: Without Exiting patterns in FAST A output

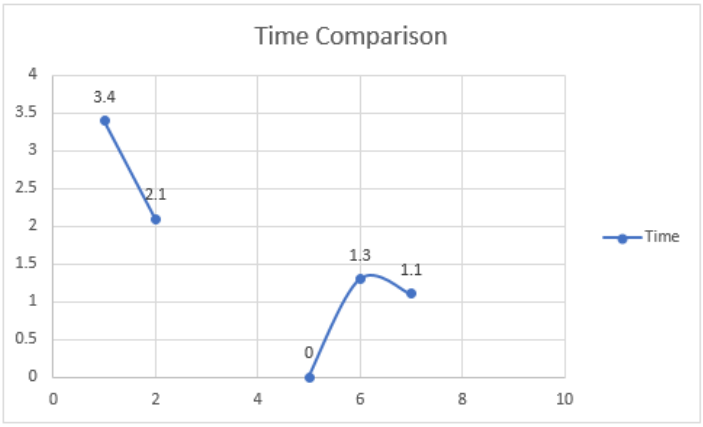

Figure 10: Without Exiting patterns in FAST A output

With existing patterns in FAST A output	
DNA String size 3	Time
Run 1	12.55
Run2	13.15
Run 3	14.58
Run 4	11.62
DNA string size 4	Time
Run 1	39.44
Run 2	44.86
Run 3	39.42
Run 4	55.21

Figure 11: With Exiting patterns in FAST A output

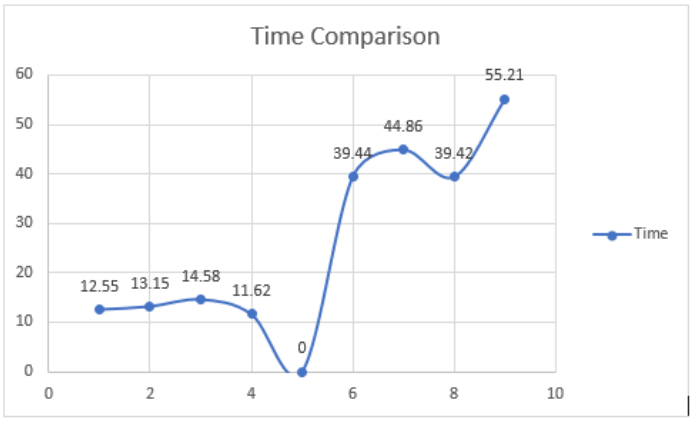

Figure 12: With Exiting patterns in FAST A output

Without existing patterns in FAST A output			
RNA String size 3	Time		
Run 1	0.3		
Run2	0.1		
RNA string size 4	Time		
Run 1	0.1		
Run 2	0.2		

Figure 13: With Exiting patterns in FAST A output

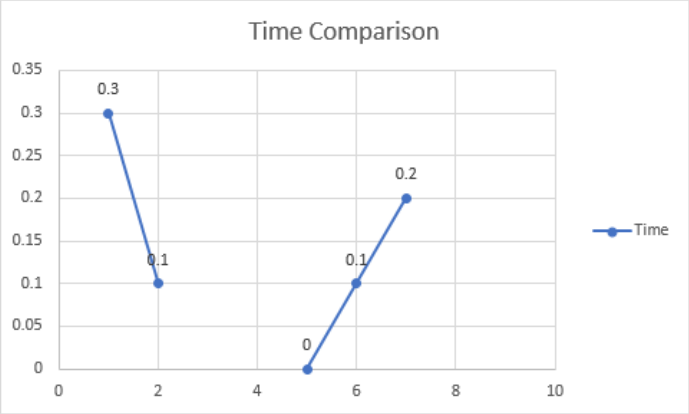

Figure 14: With Exiting patterns in FAST A output

6 Conclusion

To conclude this research, the outputs have been studied and it was found out that pattern mapping is possible. The patterns that were mapped were in small strings to match the confined scope of this research. The time taken for processing these in both DNAs and RNAs have been noted down. To conclude this, we can say that the data generate in RNAs can be mapped for pattern similarity and the number of times it exists faster than in DNA due to the vast size difference. But, as the size of DNA is more, its output is more precise and definitive as compared to RNA whose number of samples based on such small reports would require more inputs. As seen in the results section:

Pattern mapping in DNA took: 14 seconds in the first run, 12 seconds in the second run and 41 seconds in the final run where the string size was increased to 4 characters Pattern mapping in RNA took 3 seconds in the first run, 1 second in the second run and 4 seconds with patterns in 4 character strings. These outputs prove that when it comes to mapping patterns which are existing in the DNA, the string size effects the output time. But in case a string does not exist in the fasta output, the RNA returns are much quicker in proportion to the DNA ones. This means that if a string exists in the output, it would take time, but if a string does not exist, the outputs will be quick. This kind of an approach is a breakthrough as it would help us avoid datasets with no matches without even having to actually process them and letting the code just parse them at a

glance for any matches, making disease and mutation detection not only quick but also more effective.

7 Future Work

This research deals with a topic very vast and demanding. The amount of data generated till date is still being processed and even more high definition ways to analyze it is under processing. This means that working on a small part of such a big data set. For someone scaling this research, one thing that has to be kept in mind is that in case the patterns matching as symptoms for small datasets are common, larger input patterns might be required for higher definition of the data outputs. Along with that, not just strings as in this case, but also arrays with rows and columns might constitute of the patterns.

Another important point is that there could be more to the patterns than just mutation or disease detection. This kind of analysis which is done based on genetic coding opens new doors towards the industry of decoding the aspects of infants in terms of professional abilities and athletic abilities. Although the results of extending this research are endless and the prospects are uncountable, one thing has to be kept in mind that the results which specifically are linked to this research are from a smaller dataset that can just highlight the usage of this research in the real world.

Acknowledgements

I would specially like to thank my supervisor Prof. Victor Del Rosal for helping me and guiding me through the whole research and implementation process . Without his guidance and advice it would not have been Impossible to achieve this milestone. He always guided me regarding the flow structure and the relevant information and related topics to complete this research

References

Han, J., Haihong, E., Le, G. and Du, J. (2011). Survey on nosql database, *Pervasive computing and applications (ICPCA), 2011 6th international conference on*, IEEE, pp. 363–366.

Langmead, B., Trapnell, C., Pop, M. and Salzberg, S. L. (2009). Ultrafast and memory-efficient alignment of short dna sequences to the human genome, *Genome biology* **10**(3): R25.

Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., Marth, G., Abecasis, G. and Durbin, R. (2009). The sequence alignment/map format and samtools, *Bioinformatics* **25**(16): 2078–2079.

Livak, K. J. and Schmittgen, T. D. (2001). Analysis of relative gene expression data using real-time quantitative pcr and the 2- $\delta\delta$ct method, *methods* **25**(4): 402–408.

McKenna, A., Hanna, M., Banks, E., Sivachenko, A., Cibulskis, K., Kernytsky, A., Garimella, K., Altshuler, D., Gabriel, S., Daly, M. et al. (2010). The genome analysis

toolkit: a mapreduce framework for analyzing next-generation dna sequencing data, *Genome research* **20**(9): 1297–1303.

Muller-Funk, U., Thonemann, U. and Vossen, G. (2009). The munster skeleton library muesli–a comprehensive overview, *ERCIS Working Paper*, number 7.

Patel, R. K. and Jain, M. (2012). Ngs qc toolkit: a toolkit for quality control of next generation sequencing data, *PloS one* **7**(2): e30619.

Rothberg, J. M. and Leamon, J. H. (2008). The development and impact of 454 sequencing, *Nature biotechnology* **26**(10): 1117.

Seitan, V. C., Faure, A. J., Zhan, Y., McCord, R. P., Lajoie, B. R., Ing-Simmons, E., Lenhard, B., Giorgetti, L., Heard, E., Fisher, A. G. et al. (2013). Cohesin-based chromatin interactions enable regulated gene expression within preexisting architectural compartments, *Genome research* **23**(12): 2066–2077.

Van Berkum, N. L., Lieberman-Aiden, E., Williams, L., Imakaev, M., Gnirke, A., Mirny, L. A., Dekker, J. and Lander, E. S. (2010). Hi-c: a method to study the three-dimensional architecture of genomes., *Journal of visualized experiments: JoVE* (39).

Wingett, S., Ewels, P., Furlan-Magaril, M., Nagano, T., Schoenfelder, S., Fraser, P. and Andrews, S. (2015). Hicup: pipeline for mapping and processing hi-c data, *F1000Research* **4**.

Yaffe, E. and Tanay, A. (2011). Probabilistic modeling of hi-c contact maps eliminates systematic biases to characterize global chromosomal architecture, *Nature genetics* **43**(11): 1059–1065.

Zikopoulos, P., Eaton, C. et al. (2011). *Understanding big data: Analytics for enterprise class hadoop and streaming data*, McGraw-Hill Osborne Media.

8 Appendix

```java
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.TextInputFormat;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;
import org.apache.log4j.BasicConfigurator;
import org.apache.log4j.Logger;



public class DistinctUserDriver {
    final static Logger logger = Logger.getLogger(SODistinctUserMapper.class);

    public static void main(String[] args) throws Exception {

        BasicConfigurator.configure();
        logger.info("starting driver");



        Configuration conf = new Configuration();
        String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();

        conf.set("fs.defaultFS", "hdfs://localhost:54310");
        conf.set("fs.hdfs.impl", org.apache.hadoop.hdfs.DistributedFileSystem.class.getName() );


        Job job = new Job(conf, "| Distinct Users");


        job.setJarByClass(DistinctUserDriver.class);
        job.setMapperClass(SODistinctUserMapper.class);
        //job.setCombinerClass(SODistinctUserReducer.class);
        job.setReducerClass(SODistinctUserReducer.class);

        //&&&&&&&&&&&&&&&&&&&&&&&&
        //job.setMapOutputKeyClass(Text.class);
        //job.setMapOutputValueClass(NullWritable.class);

        //^^^^^^^^^^^^^^^^^^^^^^^^^

        job.setOutputKeyClass(Text.class);
        //job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(IntWritable.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        FileInputFormat.addInputPath(job, new Path("hdfs://localhost:54310/user/hduser/Akshay"));
        FileOutputFormat.setOutputPath(job, new Path("hdfs://localhost:54310/user/hduser/Akshay/output"));

        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

Figure 15: Code Execution

```java
import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Mapper.Context;
import org.apache.log4j.Logger;
import org.slf4j.LoggerFactory;



public class SODistinctUserMapper extends Mapper<LongWritable, Text, Text, IntWritable> {
    public void map(LongWritable key, Text value, Context context)
            throws IOException, InterruptedException {

        String line = value.toString();

       // String [] items =Line.split("");
        char[] carr = line.toCharArray();
        for (char c : carr) {

            System.out.println(c);
            context.write(new Text(String.valueOf(c)), new IntWritable(1));
        }

    }

}
```

Figure 16: Code Execution

```
|
public class TimeTest1 {
    public static void main(String[] args) {

          long startTime = System.currentTimeMillis();

          long total = 0;
          for (int i = 0; i < 10000000; i++) {
             total += i;
          }

          long stopTime = System.currentTimeMillis();
          long elapsedTime = stopTime - startTime;
          System.out.println(elapsedTime);
       }

}
```

Figure 17: Code Execution