Abstract

This paper touches on several different aspects in the field of bioinformatics. Some of the challenges facing the bioinformatics community will be discussed along with why these challenges still need a lot of research. How has this changed the last few years with the respect to the Next-Generation Sequencing tools that have become available. There has been done a lot of research the last few year on programs to handle the increase in amount of data generated with the NGS tools. But has this been succesful or are the biggest breakthroughs still a head? How can the relatively new programs be improved to deal with the varying problems that occur when dealing with these types of data. The results from the initial research will be used to describe how an automated system can improve performance and user friendlyness of quite advanced programs. And how this can help researcher when working on data that can not be proven correct.

Contents

Chapter 1

Introduction

This research project has several goals. First there shall be conduct an analysis of different technologies used in Next-Generation Sequencing (NGS) and sequence alignment. This initial research is also ment to give the participant a better understanding about the field of bioinformatics, molecular- and computational biology. Understanding the different challenges is important to understand why this research is important. Without the basics in place, the second part of this research project will be difficult to comprehend. The second part of this project is to create an automated, efficient pipeline for analysing NGS data. The pipeline should be constructed based one the results gathered in the first part of this project, namely what is the most efficient way of doing this NGS analysis, and what has to be done to fully automate the process. The finnished product could potentially increase the productivity of biologist working with these types of data.

1.1 Motivation

The motivation for this research project is the "Next-Generation Sequencing" (NGS) technologies that have been developed the last few years. NGS are varius methods for generating short DNA/RNA sequences from biological samples. These technologies vastly increase the amount of data, per hour, produced with respect to the older technologies. The increase in data has resulted in a bigger need for computational power and algorithmic efficiancy simply to handle all the information available. This has again resulted in a lot of research being done on developing new methods for analysing all this data. With all the new mapping/alignment tools developed the last few years, it's a big challange for researchers to know what tools to use for differents types of data. Another problem is understanding the results produced by the different tools and what results can be expected considering the different data being analysed.

1.2 Problem

The goal of this project is to study existing methods for mapping short sequences to an existing genome and design, implement and test a pipeline for identifying small non-coding RNA genes from large scal sequencing experiments.

Chapter 2

Background

2.1 Bioinformatics

Bioinformatics in general is to apply statistics and computer science to molecular biology. Traditionally this has focused on management and analysis of biological data. Using computer science to manage biological data is a extremely important part of molecular biology today. The research in molecular biology being done today would not be possible if not for the increased computational power and storage ability of computers. Still, computers are one of the bottlenecks of computational biology.

2.2 Sequence alignment

Sequence alignment is a computational method of aligning sequences to one another for comparion, aligning unknow sequences to know references to retrive information about them or aligning short sequences to a reference genome to learn more about their functions. The latter part is the most relevant for the project at hand. The reference genome is huge data set containing the DNA sequence of all chromosomes in the human body. The short sequences we want to align to this reference genome is, in this case, a huge amount of 35bp long DNA sequences. There are several different methods for computing this alignment, some more realistic than others. For instance, with the amount of data we are processing in these experiments, using a naive method by checking every position in the short sequence against every position in the reference genom, will not complete in reasonable time. So there is a great need for faster methods for doing this alignment.

Chapter 3

Tools and Methods

This chapter will describe some of the popular tools used in the analysis of sequence data, and what methods or algorithms these tools use. This analysis will be done by atempting a complete mapping with the respective tools. The data that will be used in these experiments are 10 different, but similar, data sets of about 28 million short DNA sequences. Figure 3.1 shows a small sample from a sequence file. These data sets have been made available on the server the experiments will be executed on. The mapping of these sequences will be done to the human genome, which also is available on the same server.

3.1 Next Generation Sequencing

It is important to mention that there are several different methods for producing the sequence data. The different methods also produce different types of data. For instance the illumina [12] platform produces "nucleotide space" sequences. This means it uses the alphabet A, C, G, T to represent the respective nucleotides. The SOLiD [1] platform from Applied Biosystems is not quite as straight forward as it produces sequences in "color space". This means it represents the nucleotides as numbers in the sequence, and each number represents a transition between nucleotides, not just a single nucleotide. Figure 3.2 is an illustration from [10] that shows the transitions between nucleotides in color space. In [3] there is done an thorough analysis of some of the more popular NGS platforms.

3.2 Algorithms

As mentioned earlier, the need for higher computanional efficiancy is a key factor in the development of new bioinformatic tools the last few years. But increasing the speed of computers is not a very economic way of increasing the computational power. Doubling the computers speed will only half the computation time, and if you need a speedup of x10 or even x1000, bying this in

```
>1_33_17_F3
T2223212331021020113031113231200111 1
>1_33_42_F3
T0211130130120023220013302012303211 1
>1_33_539_F3
T2310201030310102012001111221010111 0
>1_33_666_F3
T0323200321321212311012103022010003 1
>1_33_730_F3
T1132332300313220031111111011110211 0
>1_34_131_F3
T3121200133223012332203130313112112 0
>1_34_1069_F3
T2320203020312232210303001221221101 3
>1_34_1114_F3
T0323203320011212311011100122011001 1
>1_34_1156_F3
T3112031120103302013333331122320021 1
>1_35_216_F3
T2202221123123111330201030311112312 0
>1_35_467_F3
T3232323212332123110333031120113131 3
>1_35_891_F3
```

Figure 3.1: Example of sequence data

	A	C	G	T
A	0	1	2	3
C	1	0	3	2
G	2	3	0	1
T	3	2	1	0

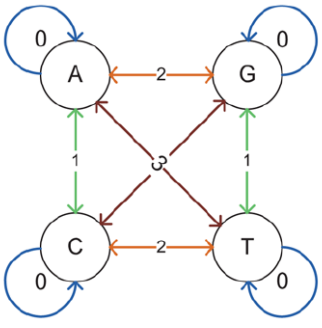

Figure 3.2: Illustration from [10]

purely computer power is not feasable. So developing new and faster algorithms is the best way to cope with the growing demand for efficiancy. In fact, there has been introduced several new tools the last years that directly approach this problem.

3.2.1 Needlman-Wunsch

The Needleman-Wunsch [9] algorithm (NW) was published in 1970, and it was one of the first major breaktroughs in computational biology. It was developed by Saul B. Needlman and Christian D. Wunsch. The NW algorithm is a classical example of dynamic programing, and was the first dynamic programing algorithm used in computational biology. This algorithm is designed to do a so called global alignment between two (or more) given strings, meaning that it will find the best possible alignment from start to end of all strings. In fact, there may be more then one "best" alignment. In the most naive sence, the best alignment is the number of exact matches between the given strings.

3.2.2 Smith-Waterman

The Smith-Waterman [11] algorithm (SW) is a variation of the NW algorithm. If we say NW is designed to do global alignment, the SW algorithm is designed to do local alignment. It differts fro NW by ignoring the parts of the strings that do not match or reduce the score of the best alignment.

3.2.3 Burrows-Wheeler Transformation

The Burrows-Wheeler Transformation [2] algorithm (BWT), also called block-sorting compression, is a data compression algorithm. It was invented by Michael Burrows and David Wheeler in 1994. It works by taking in a target string, and changing the permutation of the string so that the output string has more streches of equal characters, whitch again makes the compression easier. The rearangement is done by "rotating" the string n times, n being the length of the string. If the original string had several substrings that occur often, then the output string will have several places where a single character is repeated multiple times in a row which enhaces the compression. This makes the BWT technique very usefull in bioinformatics because of the relative small alphabet and large sequences. Figure 3.3 shows the input and output of the BWT algorithm, the transformation itself is not very interesting.

3.3 Programs

The programs ment for testing are MAQ [7], bwa [5], bowtie [4], SHRiMP [10] and soapv1 [8]. All these programs will be presented in more detail later.

Figure 3.3: BWT I/O

3.4 Hashing the genome

3.4.1 SOAPv1

SOAP [8] uses Illumina-Solexa reads for alignment. The program loads the reference sequence into the memory and creates a hash table for seed indexing. The index requires 2 bit per bp, so it can store 4 bp's per byte. This gives a quite big, but predictable memory footprint (depending on the size of the reference genome). It uses a look up table to keep track of the number of mismatches between the reference and the reads. The program searches for identical hits first, then with 1 mismatch, then 2 mismatches before it searches for gapped alignments. This allows for both gapped and ungapped alignments. SOAPv1 appears to only align nucleotide space sequences, so it can not be used to map the SOLiD reads for this project.

3.4.2 Others

There are several other tools the use the method of hashing the genome, including PASS (Campagna et al., 2009), MOM (Eaves and Gao, 2009), ProbeMatch (Jung Kim et al., 2009), NovoAlign (htto://www.novocraft.com), ReSEQ (http://code.google.com/p/re-seq), Mosaik (http://bioinformatics.bc.edu/marthlab/Mosaik) and BFAST (http://genome.ucla.edu/bfast).

3.5 Hasing the query

3.5.1 MAQ

MAQ [7] is designed to handle Illumina-Solexa data, but it can also handle SOLiD data. Like SOAPv1, MAQ uses a two-seed pigeonhole filtering technique. It also uses the hamming distance in the filtering step. Flexible memory footprint. It is however clear after some research, that MAQ is very slow compared to BWA, Bowtie and SHRiMP, so it will not be tested.

3.5.2 SHRiMP 1

The SHRiMP [10] algorithm draws upon three recent developments in the field of sequence alignment: q-gram filter approaches, spaced seeds and specialized vector computing hardware to speed up the Smith-Waterman Algorithm. is another tool that uses a hashed queries instead of hashing the genome. Flexible memory footprint.

3.5.3 Others

There are also many other tools that use hashed queries. Eland (Cox, 2007), RMAP (Smith et al., 2008), ZOOM (Lin et al., 2008), SeqMap (Jiang and Wong, 2008) and CloudBurst (Schatz, 2009).

3.6 Compressing the genome with BWT

3.6.1 Burrows-Wheeler alignment (BWA)

BWA [5] is a alignment package base on the Burrows-Wheeler algorithm. BWA uses the BWT compression technique to build a prefix trie of the reference genome. It performs gapped alignment for single-end reads, supports pair-end mapping, generates mapping quality and gives multiple hits if required. By default the output is in the SAM format (http://samtools.sourceforge.net). It uses backwards search for exact and inexact matching. BWA implements a BFS search on a heap-like data structure. BWA can handle both Illumina-Solexa and SOLiD data.

3.6.2 Bowtie

Bowtie [4] uses a method very similar to BWA, but by default it uses a DFS. If there is no exact match, the search is not guarantee to find the best inexact match. It can applay a BFS search to quarantee the best inexact match, but that will slow it down slignificantly. Bowtie can also handle both Illumina-Solexa and SOLiD data.

3.7 Comparison

There have been done ample atempts on comparing varius alignment programs, and interpreting the results can be a bit tricky. The authors of different articles don't use the same criteria when comparing different tools. In this section some of the results will be presented.

Programs	Time (s)	Conf (%)	Err (%)
Bowtie-32	1271	79.0	0.76
BWA-32	823	80.6	0.30
MAQ-32	19797	81.0	0.14
SOAP2-32	256	78.6	1.16

Figure 3.4: Evaluation on simulated data, Li et. al

Programs	CPU time	Wall clock time	Memory footprint	Reads Aligned
Bowtie -v 2	15m	15m	1,149	67.4
SOAP	91h 57m	91h 47m	13.619	67.3

Figure 3.5: Bowtie alignment performance versus SOAP, Langmead et. al

Programs	CPU time	Wall clock time	Memory footprint	Reads Aligned
Bowtie	17m	18m	1.353	71.9
MAQ	32h 56m	32h 58m	804	74.7

Figure 3.6: Bowtie alignment performance versus MAQ, Landmead et. al

Chapter 4

Experiments

First of all, by understanding the approaches the different tools use, it is clear that there is a need for some preprocessing of the input data used in the experiments. Both Bowtie and BWA use the Burrows-Wheeler Transformation and FM-indexing technique to create an index over the reference genome. These indices are also quite memory efficient, so they can easily fit in the memory of the test server used in the experiments. This indexing is not done in real time as the mapping is executed, but the indices have to be created manually before the mapping is done. SHRiMP however uses another indexing technique, by indexing the reference genome into spaced seeds. This results in an index that is too big too fit in memory, so the SHRiMP package provides a tool to split the index into smaller chuncks that fit in the memory at execution time.

An important factor in working with the given data set is that, it is real, unprocessed data, so there is no accurate reference on what results can be expected performance wise.

4.1 Initial alignment

The first alignment was performed with all three programs running on default settings. As indicated by [5], [4] and [10] this was not the optimal configuration, but it would hopefully give a pointer on their performance compared to each other. The tests were done sequential on the same server. The results where expected to be inferior to the results presented in 3, but other than that, there were no expectations. When all the programs had performed the alignment, the results were analysed, and this seemed very odd at first. All the tests had aligned less then 5% of the sequences to the reference genome. It was indicated that an adaptor sequence was added to all the 35bp sequences by the NGS technology used. There were done several atempts to locate the adaptor sequences with no luck. A new approach was needed to complete the alignment with more or less valid results.

4.2 Refined alignment

After some more research about the adaptor sequence that supposedly gave the undesirable results, it was clear that one way to avoid this problem was to use the "trim" function of Bowtie. This alows the user to specify a number of nucleotides to trim away from all the sequences being mapped to the reference genome. There were done several trials on small subsets of the sequences to see if trimming the end presumed containing the adaptor would increase the alignment. The test confirmed that this was the case, and that trimming 13 nucleotides off the end gave the most reliable results. This ment that the 35bp sequences were reduced to 22bp, which is close to the expected length of an mRNA.

Chapter 5

Implementation

This chapter will go more into detail on the process of creating a pipeline. When the method of choice has been refined to give presumed accurate results it is time to start automating the alignment and analysis process. The idea of a pipeline comes as a result of the complexity and time consuming process of manually doing this analysis. Having an automated pipeline is also a good way to standardize the whole process. While manually performing 4-5 advanced steps in a complex process easily can be error prone, an automated pipeline configured with a predefined parameters always will performe the same task the same way.

5.1 Galaxy pipeline

The "Galaxy framework"[13] has been used to create the pipeline for finding explicitly expressed miRNA's. "The Galaxy Framework is a set of reusable software components that can be integrated into applications, encapsulating functionality for describing generic interfaces to computational tools, building concrete interfaces for users to interact with tools, invoking those tools in various execution environments, dealing with general and tool specific dataset formats and conversions, and working with metadata describing datasets, tools, and their relationships. The GALAXY Application is an application built using this framework that provides access to tools through an interface (e.g., a web-based interface). A GALAXY Instance is a deployment of this application with a specific set of tools."

Creating a pipeline, for anything really, could seem like a very difficult and time consuming job. But the design of the Galaxy framework makes this much easier. What Galaxy does for you is to create all the conections and handle the logistics so you can focus on the more interseting parts that go into the pipeline. Figure 5.1 shows a screen shot from the Galaxy interface. This shows how the modules are conected, what types of I/O is going trough the pipeline and what tools are performing each step. Creating this is just a matter of drag and drop.

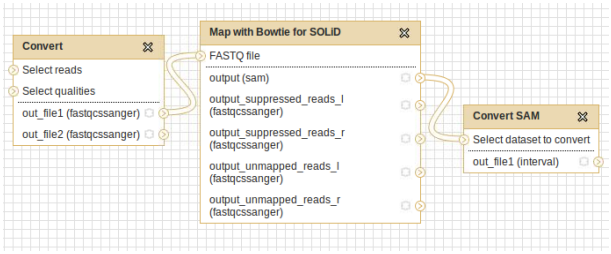

Figure 5.1: Screen shot from Galaxy workflow

All the tools included in Galaxy is found in a simple menu on the left side in the interface and can be draged into the workflow window. Configuring the modules is also done from this screen.

5.1.1 First step

The first step is a operation included in the Galaxy framework. It is simply called “Convert” and can be found under “NGS: QC and manipulation” in the web interface. It takes two files as input, “input.csfasta” and “input.qual”, and converts these into a “output.fastq” file. This is neccesary because Galaxy interfaces with bowtie a little more restrictive then you have to when performing the mapping manually. This step could also have been done manually, but was disregarded. With Galaxy, there is simply no option.

5.1.2 Second step

The second step is the mapping with bowtie. It is found under “NGS: Mapping” in the interface and is called “Map with bowtie for SOLiD”. The bowtie tool is not included in the Galaxy framework, so it has to be pre-installed on the computer running the Galaxy server. There is also some configuration needed to make Galaxy aware of the location and type of the pre-built bowtie index. The Galaxy framework includes configurations for all the tools it supports, so there are only small changes that is needed to make everything work together. The configuration for the bowtie index is found under “tool-data” in the Galaxy folder. In this case with a color-space index, it is the “bowtie_indices_color.loc” that needs some altering. The file itself contains explenations and instructions on how to configure it. When the index has been added to the Galaxy interface, bowtie is ready to do the mapping. However, the default configuration of bowtie in the Galaxy interface is not very useful for this pipeline. All the experiments presented earlier shows a very specific configuration of the parameter needed to get usefull results on these data sets. Similar to the input to bowtie, the output is also restricted to only one format, namely Sequence Alignment/Map format (SAM) [6].

5.1.3 Third step

The third step is just another convertion. The sam file contains a lot redundant information for the intended purpose, so this step is just simplyfying the file by removing several of the columns containing unnccesary information. The tool that is used is a part of the "SAMtools" package mentioned breifly above. It is called "Convert SAM" and is found under "NGS: SAM Tools" in the Galaxy interface.

The first three steps are the only once that actually is implemented in the pipeline so far. The next step uses the output from the pipeline, but also requires another data set that can be accuired through the Galaxy interface, but it can not automatically be inserted into the pipeline that is configured. All the input data has to be specified before starting the operations in this pipeline, and data that is required in the middle of the pipline simply can not be added at the start. This is a major limitation of the Galaxy framework with no apparent work-around at this point.

Chapter 6

Discussion

Here I will discuss some of the results gathered in this project. Most of what has been accomplished is valueable insigth into the field of bioinformatics. How different choices have a great impact on the results of your work. That there are many possible solutions to almost any problem, and diving into a problem with no expectations may effect the outcome of your research.

6.1 Analysing the results

As mentioned in 4 the results of the initial experiments where quite odd. But by using the method of trimming the end of the sequences, some interesing results were found. It was also mentioned in 4 that there is "no accurate reference on what results can be expected performance wise". So using information from other similar test is the best way to verify that your results have some validity.

There were done several small scale test on the data sets with bowtie [4]. This was mainly to get a better understanding of the results. The small scale tests aimed to find out what impact it would have to change the trim length when mapping the sequences. Hopefully the same sequences would map to the same region of the genome all the time. This was inconclusive, but another comparison was done regarding what sequences was mapped when adjusting the trim length. The overlapp in the results were far below what I expected. For instance, the overlap between the 10 nucleotide trim and the 15 nucleotdei trim is only about 1%. The overlapp between 12- and 13 nucleotide trim were better, about 60+% of the sequences in "12" where also found in "13". Though this is better, it's still less then I had expected. Even though the results where quite surprising, it does not nessecarily mean it's bad. It could mean that altering the parameters will help find a lot more sequences then just using the "optimal" parameters.

6.2 Future work

The pipeline was not satisfactory completed, partly because of the limitations in the Galaxy framework [13]. One possible solution is to create a whole new module that can work around the current problems of Galaxy. Another possibility is to create a seperate solution that can interact with Galaxy to eliminate the problem in stead of working around it. A third option is to create a completely independant solution, which actually was the first idea for this project. This however is a much bigger challange, and it will most likely be harder to maintain and to operate than the interface of Galaxy.

Chapter 7

Conclusion

To summarize, this project has discovered alot when it comes to the process of working with NGS data and alignment programs developed to handle this type of data. The results were affected by the fact that two of the alignment programs used, namely BWA [5] and SHRiMP [10] could not handle the data sets used. This left only one option, which is better than no alternative, but cut some of the research short because the programs could not be compared satisfactory. There is also a chance that the data sets are not optimal, that the quality/correctness of the sequence data is not very good. In that case how could this have been avoided? Could I have aquired other sequence data that is known be more "correct"? When it comes to the second part of the project, the contruction of a pipeline for mapping next generation sequence data. This could have been handled very different. The limitations of Galaxy [13] could have been discovered a lot earlier, and another soultion could have been constructed, or the problems could have been worked around. But in the end, all the knowledge gained from this project is something I will take with me and hopefully the results I have gathered can be usefull for future work.

Bibliography

[1] Applied Biosystems. Solid. http://www.appliedbiosystems.com/absite/us/en/home/applications-technologies/solid-next-generation-sequencing/next-generation-systems.html.

[2] Michael Burrows and David Wheeler. A block sorting lossless data compression algorithm. *Digital Equipment Corporation*, 1994.

[3] Olivier Harismendy, Pauline C Ng, Robert L Strausberg, XiaoyunWang, Timothy B Stockwell, Karen Y Beeson, Nicholas J Schork, Sarah S Murray, Eric J Topol, Samuel Levy, and Kelly A Frazer. Evaluation of next generation sequencing platforms for population targeted sequencing studies. *Genome Biology*, 10(3), 2009.

[4] Ben Langmead, Cole Trapnell, Mihai Pop, and Steven L. Salzberg. Ultrafast and memory-efficient alignment of short dna sequences to the human genome. *Genome Biology*, 10(3), 2009.

[5] Heng Li and Richard Durbin. Fast and accurate short read allignment with burrows-wheeler transform. *BIOINFORMATICS*, 25(14):1754–1760, 2009.

[6] Heng Li, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, and Richard Durbin. The sequence alignment/map format and samtools. *BIOINFORMATICS*, 25(16):2078–2079, 2009.

[7] Heng Li, Jue Ruan, and Richard Durbin. Maq: Mapping and assembly with qualities. *Genome Research*, 18(11):1851–1858, 2008.

[8] Ruiqiang Li, Yingrui Li, Karsten Kristiansen, and Jun Wang. Soap: short oligonucleotide alignment program. *BIOINFORMATICS*, 24(5):713–714, 2008.

[9] Saul B. Needlemana and Christian D. Wunscha. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, 1970.

[10] Stephen M. Rumble, Phil Lacroute, Adrian V. Dalca, Marc Fiume, Arend Sidow, and Michael Brudno. Shrimp: Accurate mapping of short color-space reads. *PLoS Computational Biology*, 2009.

[11] Temple F. Smith and Michael S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197, 1981.

[12] Solexa. illumina. http://www.illumina.com/technology/sequencing_technology.ilmn.

[13] Galaxy team. Galaxy-central. https://bitbucket.org/galaxy/galaxy-central/overview.