UPPSALA
UNIVERSITET

MSc BIOINF 13 001

Exploring the diversity of unmapped reads from human deep sequencing

Amin Zarif Saffari

Degree project in bioinformatics, 2012
Examensarbete i bioinformatik 30 hp till masterexamen, 2012
Biology Education Centre and Department of Immunology, Genetics and Pathology, Science for Life Laboratory Uppsala, Rudbeck Laboratory, Uppsala University
Supervisor: Prof. Lars Feuk

Abstract

Currently DNA and RNA sequencing are performed as standard parts of many scientific experiments. While the majority of the reads produced in these experiments do map to the genome of the organism of interest there are a significant fraction that do not. These reads have often been viewed as uninteresting and thus discarded, sometimes explained as errors created in the sequencing process. However, there may be a real possibility that these reads actually contain genomic sequences belonging to, but not currently in the genome of the organism investigated, as well as information about other organisms which live and thrive in the sample material. Considering this, it is of great interest to investigate these reads to see if they contain any usable information. In this project the unmapped reads from SOLiD sequencing of blood and saliva from a twin pair were assembled. The assembled parts were then compared to different blast databases to investigate if similar genomic regions are reported in other species. We can conclude that indeed a large fraction of the contigs found in this assembly have homology to bacterial genes while other contigs share similarity to genomic regions found in apes and other species closely related to us. All in all the results show that there is more to the unmapped reads than just sequencing errors.

Exploring the diversity of unmapped reads from human deep sequencing
-Popular science summary

Amin Zarif Saffari

THE exploration of the human genome is a crucial scientific task in the context of modern biology. One of the main goals of exploring the human genome has been to create a map of the whole of the human genome. This map was created by investigating a few individuals genomes, by first splitting the genomes in smaller parts and then carefully investigating each part and make a small map of that specific part. The smaller parts were then assembled together like a puzzle to create larger parts, until the sequence of the full human genome was revealed. This sequence is called the human reference genome.

The human reference genome quickly became an important tool for researchers around the world investigating different aspect of human biology. The knowledge of the full genome sequence gave researchers the tools to investigate the genome in a new way, giving them the ability to investigate genomic aberration in the context of their genomic coordinates.

Another important aspect of the human reference genome is that it works as a guide when investigating the data from whole genome sequencing runs. When doing a whole genome sequencing, the sequence of short parts of the genome called reads are revealed, and they are fitted to the human genome sequence. This way, small differences can be detected between individuals, and the cause of different conditions such as diseases can be revealed. However the human genome reference is not fully completed and some parts of it may not have been discovered yet. In order to try to find and to understand the content of these parts we investigated the reads that did not fit anywhere to the human genome, to see what sequences they might derive from. To aid us we used a database containing genetic sequences from a wide array of different kinds of organisms, containing both eukaryotes and prokaryotes and viruses. Our main hypothesis was that some of the sequences found may derive from either other organisms living inside our bodies, or come from sequences that do occur in the human genome, but have not yet been identified as part of the human reference genome.

After comparing the genetic material found not to fit to any known part of the human reference genome, we found a wide array of different sequences. Many of whom had a close similarity to different bacteria, and some with similarity to our closest relatives, opening up for the possibility of this strategy being a fruitful approach, both when investigating the genomes of species co-existing with us as well as finding new parts of the human genome sequence.

Degree project in bioinformatics, 2012
Examensarbete i bioinformatik 30 hp till masterexamen, 2012
Biology Education Centre and Department of Immunology, Genetics and Pathology, Science for Life Laboratory Uppsala, Rudbeck Laboratory, Uppsala University
Supervisor: Prof. Lars Feuk

...A hole had just appeared in the Galaxy. It was exactly a nothingth of a second long, a nothingth of an inch wide, and quite a lot of million light years from end to end. [...] Somewhere in the deeply remote past it seriously traumatized a small random group of atoms drifting through the empty sterility of space and made them cling together in the most extraordinarily unlikely patterns. These patterns quickly learnt to copy themselves (this was part of what was so extraordinary of the patterns) and went on to cause massive trouble on every planet they drifted on to. That was how life began in the Universe...

-Douglas Adams: The Hitch Hiker's Guide to the Galaxy (1979)

Contents

Abstract i

Popular science summary iii

List of abbreviations viii

Introduction 1

Materials and methods 3

Results 8

Discussion 14

Supplementary material 18

Source code for various scripts used in the project 18

List of figures

1 The general workflow of the method developed in this project. 3
2 Two approaches which were followed in this project 4
3 ER Diagram of project database 7
4 Assembly parameters (kmer-value/coverage-cutoff) as a function of indicators (length/N50) 8
5 Assembly parameters (kmer-value/coverage-cutoff) as a function of indicators (Contig-count/N50) 9
6 Assembly parameters (kmer-value/coverage-cutoff) as a function of percent read usage 10
7 Blood and saliva assembly similarity 10

List of tables

1 Pros and Cons of different approaches 3

2 Assembly parameter permutations of one of the saliva sample 11
3 Most frequent item on blood annotation 12
4 Most frequent item on blood and saliva annotation 13

List of abbreviations

kmer - kmer index
cvCut - coverage cutoff, measured in kmers coverage. This threshold specifies how many read k-mers must overlap for each contig kmer. The number of kmers per read is a function of read length L and k-mer length K (L-K+1). A 32bp read contains 12 21-mers. In that instance, a kmer coverage of 3.9x will correspond to a conventional X-coverage of 10x.
ctgs - number of contigs
asmLg - total length of all contigs (in base pair)
mean - mean length of the contigs (in base pair)
max - length of the longest contig (in base pair)
1k - number of contigs over 1kbp
N50- the length of the shortest contig in an assembly such that the sum of contigs of equal length or longer is at least 50% of the total length of all contigs.
tiles - number of reads that are used in an assembly
rdPc - percentage of input reads used in the assembly
HGT - Human Genome Project

Introduction

The organism's hereditary information units of all organisms is called genome which is including not only coding parts of DNA (or RNA) but also non-coding parts. A gene is constructed by a particular sets of nucleotide bases whose arrangements responsible for constructing proteins by the structural components of cells and tissues. Whole DNA in a living organism is called genome [1].

Genomics is an emerging field whereby scientists map the genetic code - the language of life - and study how differences in this language leads to differences in phenotype.Attempt to achieve the definition on the questions asked by science push forward the edge of our knowledge on Who we are and our understanding of our relationship with the environment have lead to the construction of the human genome reference.[2]

This is all made possible by a nation-wide research effort called the Human Genome Project(HGT) initiated in October 1990to grasp more information on human body (and other organisms) on the aspect of biological processes to use in medical field which could prolong our life [2] (Knowledge of the genetic code will enable us to gain a better understanding of how our genes affect our health).

With the publication of the first draft human genome the estimation of the number of human genes was estimated to be 20,000 to 75,000 [3, 4]. This was due to different computational methods to find genes, that is either to find distinct patterns that define a gene (start and stop codons) or comparing the sequence with the sequence of known genes statistically. Although the absolute number of human genes is still unknown it is believed that we have about 20,000 - 30,000 genes in our genome.

The Genome Reference Consortium (GRC) is responsible for maintaining the human reference genome(hg) which the latest release is GRC37 [5](sometimes refereed to as hg19 a release that are derived from the official GRCh37 release). This version contains 27478 contigs and 10^{-4} is almost its accuracy (fewer than 1 base error in 10 thousand bases) over each chromosome [1, 6] therefore the reference are regularly updated with new and experimentally confirmed regions. Assembly patches are categorized as:

1. Novel patches
2. Fix patches

Additional novel sequence which are not represented on the primary assembly are called novel patches. Additional sequence that will modify the known regions of misassembly in previous assembly when the next major assembly update is released are called Fix patches [7].

Recent human genome studies have showed a lot of new sequences which are not found in the human reference genome [8, 9]. A large fraction of these sequences can likely be explained

by errors created by the DNA sequencing protocols, repeat elements that may be difficult to map to the reference, or reads spanning splice junctions (in the case of RNA sequencing).

It is a known fact that when doing DNA sequencing a large fraction of the reads will not map to the reference genome. Considering this, it is highly possible that some fraction of these unmapped reads may derive from both functionally active and important parts of the genome currently not represented in the reference assembly.

There are several different sequencing technologies currently used, and the technologies are developing rapidly and several new technologies are on the way out on the market. For this project ABI Solid was used as main sequencing technology.

In this project we assemble the unmapped reads generated in the sequencing derived from blood and saliva of two twin pairs. The aims are to investigate if any currently unknown regions of the human genome may be identified when assembling these reads, as well as investigate the possibilities finding the contamination made up by DNA from micro-organisms in the sample. During this study we can report that we found a large amount of DNA that seems to derive from micro-organisms in the samples as well as some sequenced mapped to closely related species such as chimp.

Materials and methods

All DNA samples was derived from blood and saliva from a twin pair. The DNA was sequenced using a SOLiD instrument with a read length of 50 bp and 35 bp in forward and reverse strand respectively. The sequence data was aligned to the human reference genome using Applied Biosystem's BioScope with standard settings.
The general workflow of the pipeline developed will be explained in detail in this section (see figure 1).

Two different approached where used in the project (figure 2) as a means of finding the best method for building the the assembly. In the first approach all sample-reads are merged together and then assembled with the different configurations in the second strategy all samples are assembled individually and then merged .Each method have its own advantages and disadvantages (see table 1)

To pre-process the data for use in downstream analysis all unmapped reads were collected

Table 1: Pros and Cons of different approaches

	Advantages	Disadvantages
Approach 1	• More reads lead to better assembly • Higher N50	• Slow • Needs a lot of memory • Resorting the data
Approach 2	• Fast	• Some contigs disappeared

from the different samples and the unmapped reads were converted to SAM [10, 11] format for

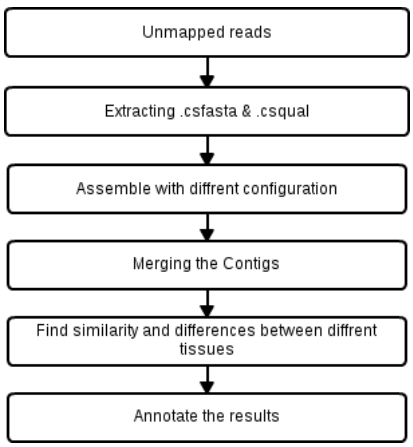

Figure 1: The general workflow of the method developed in this project.

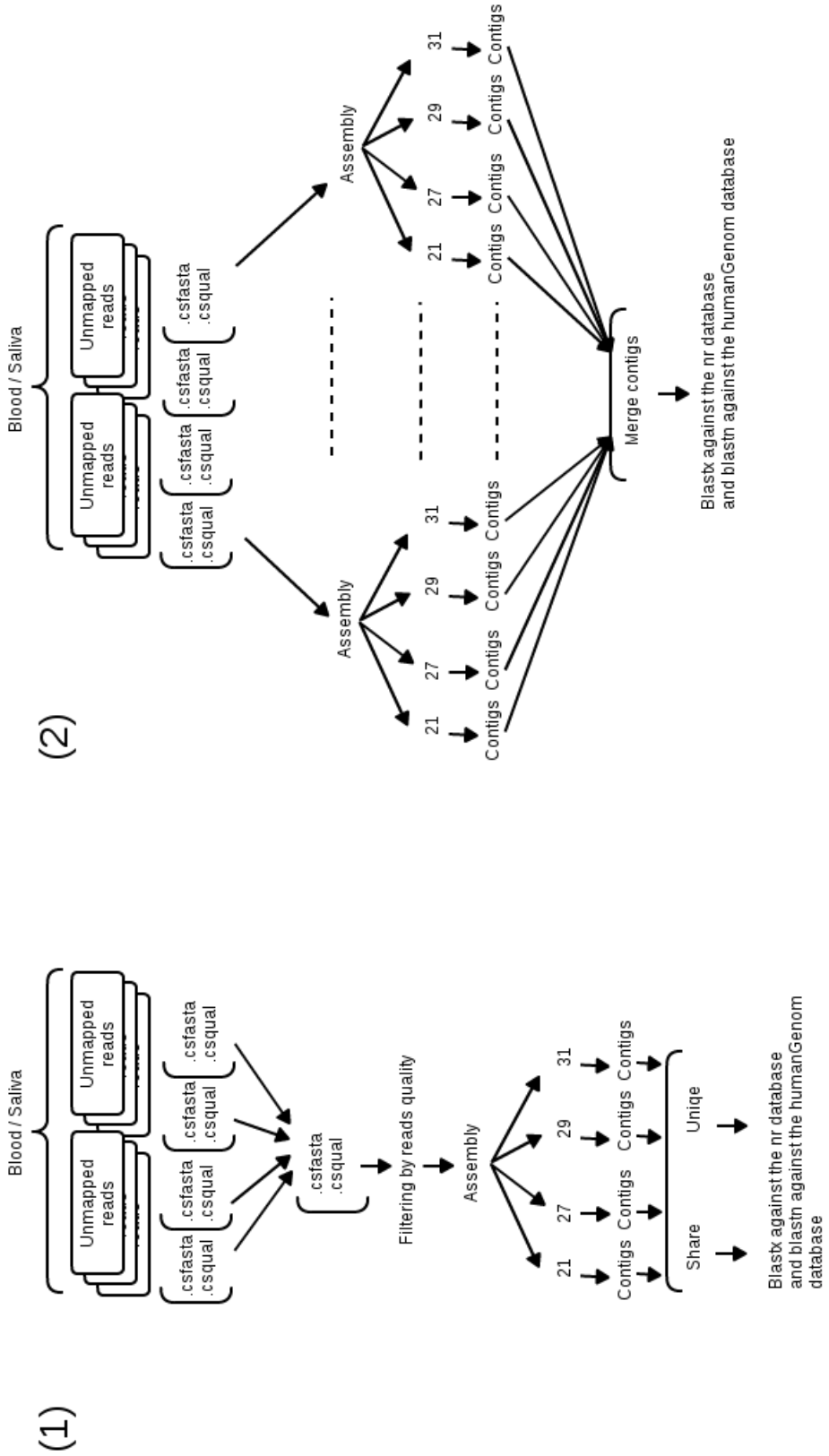

Figure 2: Two approaches which were followed in this project

use in downstream analysis (Script.1). To reduce complexity all reads having missing basecalls where discarded.
The quality values of each read was then converted to ASCII to facilitate easier downstream analysis.The phred quality score encoding method divided in these groups:

1. Sanger format: Using ASCII value 33 - 126 to encode phred quality score from "0" to "93" [10]

2. Solexa/Illumina 1.0 format: Using ASCII value 59 - 126 to encode phred quality score from "-5" to "62" [19]

3. Illumina 1.3+ format: Using ASCII value 64 to 126 to encode phred quality score from 0 to 62

4. Illumina 1.5+ format: Using ASCII value 67 to 126 to encode phred quality score from 0 to 59

The quality values in the sequence data received was evaluated and determined to be in the sanger format (see script 2). To get a quality score usable for assembly the phred quality score of each base was then calculated (see script 3).

$$\begin{aligned} mem(Kb) &= -109635 + 18977 \times ReadSize(bases) \\ &\quad +86326 \times GenomeSize(millionbases) \\ &\quad +233353 \times NumReads(millions) \\ &\quad -51092 \times K(kmer) \end{aligned}$$

Equation (1): Correlation of readSize, GenomeSize, ReadNumber and Kmer with Ram required by velvet assembler.

The probability of error was calculated as the joint probability of error in each of two consecutive color positions and as single color errors are typically correctable, the probability of a nucleotide error at a given position were calculated as the probability that both corresponding color positions are erroneous (i.e. the pairwise product of color error probabilities yields the probability of uncorrectable nucleotide errors).
A threshold of the average quality of the reads was then calculated and the reads meeting this threshold was trimmed to reduce the size and complexity of the dataset (Equation 1).
The reads was then assembled using velvet and reads with low quality was further filtered .

$$Q = -10 \times \log_{10} P$$

Equation (2): Error probability relation and base quality score[18]

Using the quality calculated by velvet (Equation 2) and the average quality score threshold of >24 the base call accuracy is 99.60% and a large proportion of the unmapped reads (64%) were discarded because they did not meet the quality threshold.

For efficient assembly of the colorspace reads a slightly modified version of Velvet was used to create the *de Brujin* graphs.When using the velvet implementation of *de Bruijn* graphs the most important points considered were [12]:

1. there is a one to one relationship between the *Bruijn* graph and the sequence and K-mer size which is given.

2. Converting the *de Bruijn* graph to the sequence didn't conclude to particular sequence and most of the time it lead to many sequences.

3. The probability of resolving *de Bruijn* graph to a particular sequence increase by longer K-mers size.

4. The maximum value of K-mer size is completely based on the amount of computer resources. Because most of the *de Bruijn* graphs are stored and processed in RAM (see Equation 1)

Velvet was used to assemble the reads using several different configurations. The kmer value and coverage cutoff parameters are the two user-defined parameters in Velvet assembly and was decided by the length, specificity and total number of contig which we want by assembly [13]. To avoid mis-assemblies the expected coverage variable was set to a low setting so as to avoid to create mis-assemblies. To put the kmer effectiveness in nutshell there is a trade off between assembly length and N50 depends on the K-mer value, for instance choosing lower value lead us to very large and fragmented assembly because of higher chances for reads to overlap by each other. On the other hand high K-mer value result us to tight and short assembly length with a higher N50 [13, 14].Assemblies was run using the Vmatch software [15] and ClustDB [16] to merge all Blood and Saliva contigs individually to cluster all the contigs which are similar, non-redundant set of contigs are created.

The suffix tree [17] algorithm was then used to find the similarity between 113 million (Blood) and 510 million (Saliva) contigs and then blast them against the database nrBlast, humanGenome and other-genomic database to annotate them and find validated sequences, here defined as sequences having homology to known proteins in other species.

To facilitate easy access of the huge amount of contigs created as well as to get a system for inter assembly comparison a MySQL database was designed to store all the data found. The database were used not only to store and search the data in a high throughput manner but also to make it easy to look up single contigs for the purpose of exploring the data. The database were built using several table, each table storing a specific piece of information (figure 4).

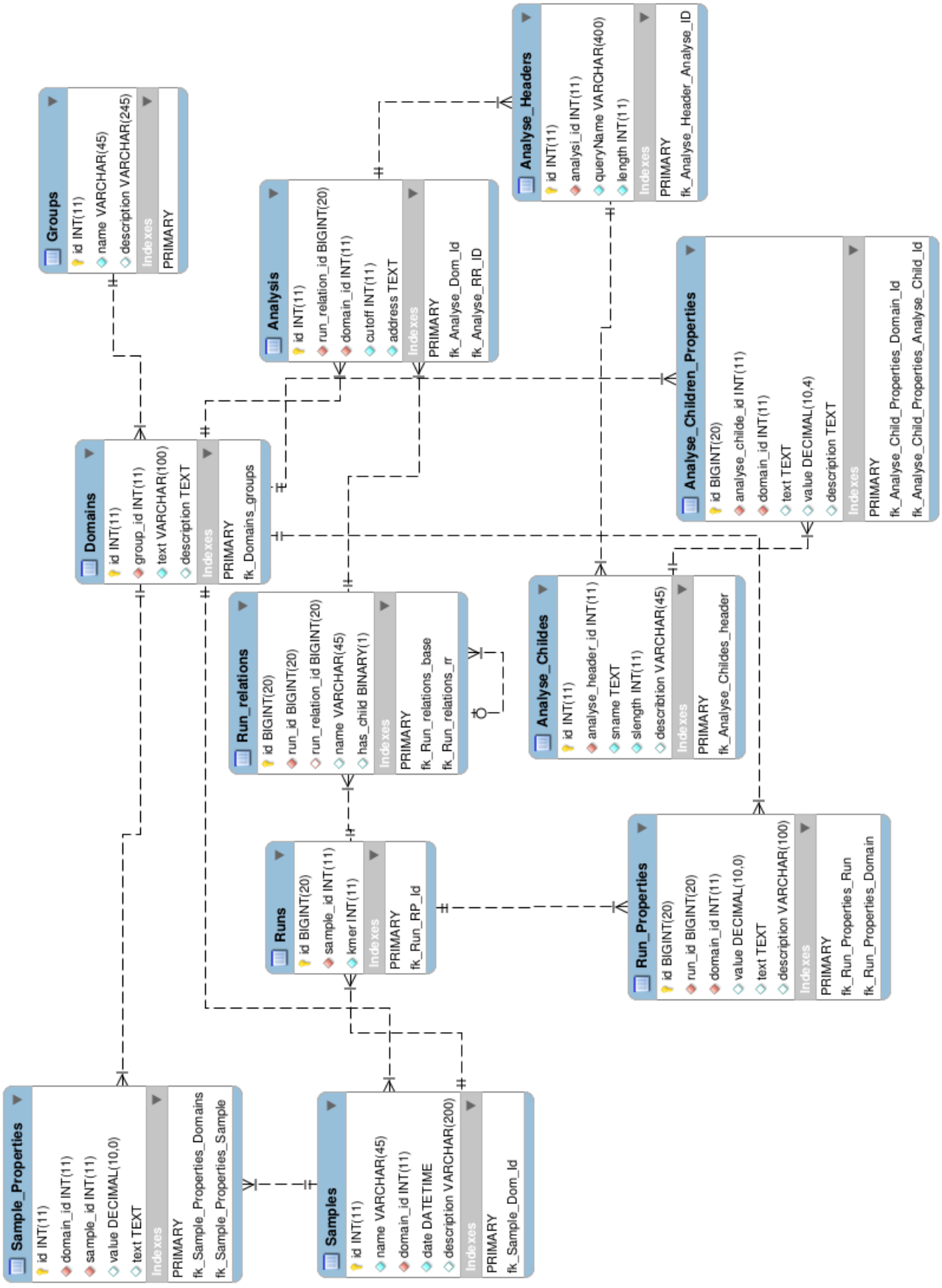

Figure 3: ER Diagram of our database

Results

One important factor of any assembly is the N50 value which basically is the weighted median contig length in the assembly where 50% of all of the contigs in the assembly have an equal or larger length than this value. Another important factor is the cvCut (coverage cutoff, measured in kmers coverage. This threshold specifies how many read k-mers must overlap for each contig kmer. The number of kmers per read is a function of read length L and k-mer length K (L-K+1)) . This value will of course differ with different run statistics and we concluded that different combinations of kmer and cvCut lead us different N50 and assembly length (see figure 4). We find that (see table 2) when using different kmer and cvCut values we get different N50 and assembly length. As seen the fraction of reads used in the assembly will increase with a lower N50 value, but the most important factor seems to be the cvCut, where a lower value gives more mapped reads (see figure 4). The auto setting typically gives a low value.

Meanwhile there is a positive correlation by using more reads (choosing lower K-mer value)

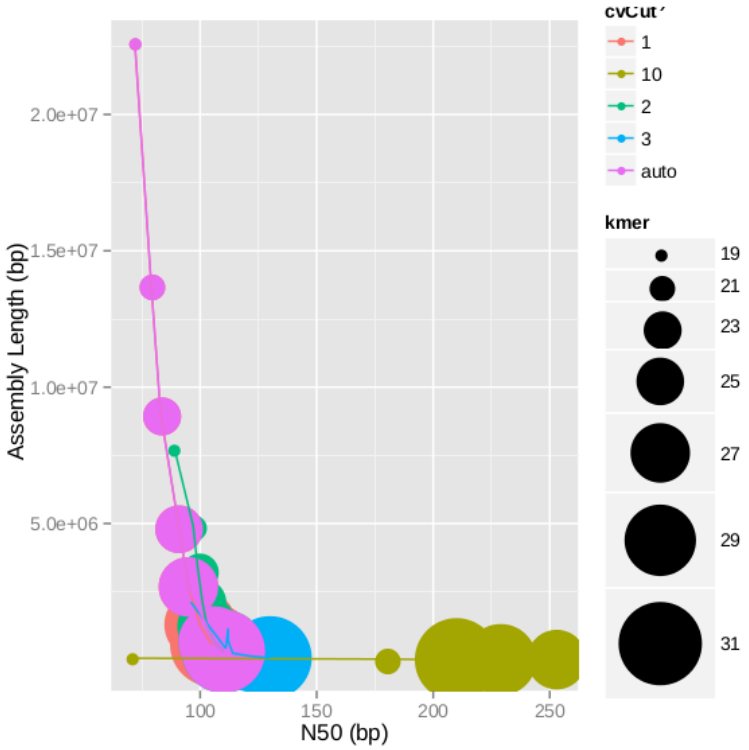

Figure 4: Assembly parameters (kmer-value/coverage-cutoff) as a function of indicators (length/N50)

and more accurate parameters which lead assembly to mis-assemblies at lower accuracy [14].To complete point above, we should consider that "every coverage cutoff" is also practical for

instance the percentage of read usage by assembly is higher at a coverage cutoff of 2 than at 3 on the particular K-mer value (see figure 6) [13, 14].

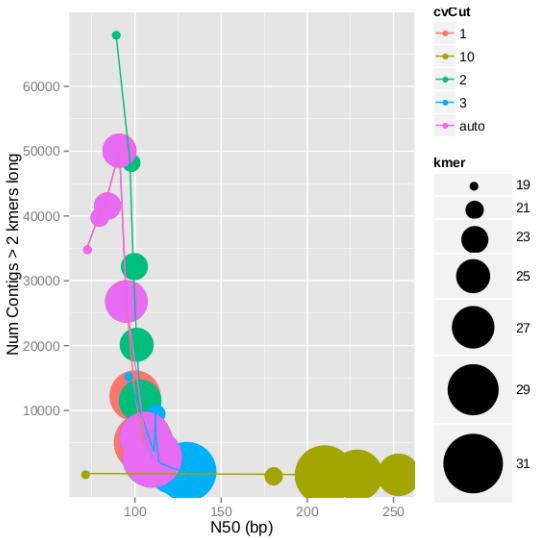

Figure 5: Assembly parameters (kmer-value/coverage-cutoff) as a function of indicators (Contig-count/N50)

This ambiguity tell us there is a "contige read trap" between coverage cutoff and length of contig, which the longer contige bounded by the shorter contigs in lower coverage. To wash this limitation out we can increase the coverage cutoff value to see longer and more acceptable contigs by allowing more reads to play role into the assembly [14].

Another important factor when doing assembly are the number of contigs found, however, as some contigs are very short (just a few base-pairs) we used a cutoff of contigs less than 2 kmers when investigating this factor. It is interesting to note that the number of contigs do not have the same relationship to the N50 and cvCut as the assembly length have (see figure 5). When merging the contigs in the blood from the both of the twins used in this project using the suffix tree algorithm we got about 113 million non-redundant contigs. Among these almost 1/4 of the contigs where found to have homology to known sequences in different species (see table 3). To investigate which of these contigs that where also found in the saliva, removing contigs belonging to DNA from micro-organism found in only one of the tissues (figure 7) as well as ruling out some of the tissue specific errors we found about 60 000 overlapping contigs where 17000 had high quality hits against the blast databases used in this project (table 4).

It is interesting to note that among these 17000 high quality contigs we find DNA found in other apes as well as DNA found in bacteria living both on the skin and inside humans.

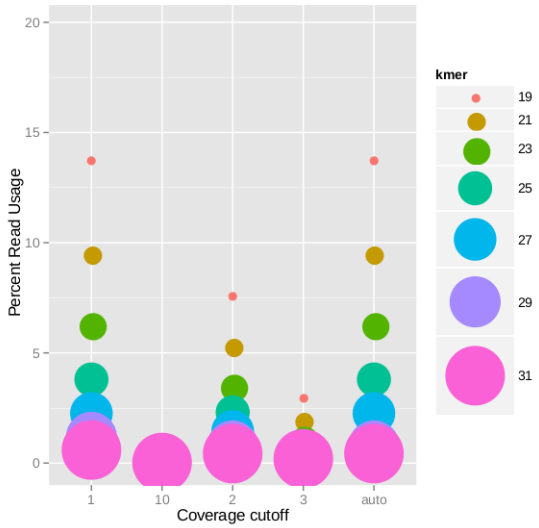

Figure 6: Assembly parameters (kmer-value/coverage-cutoff) as a function of percent read usage

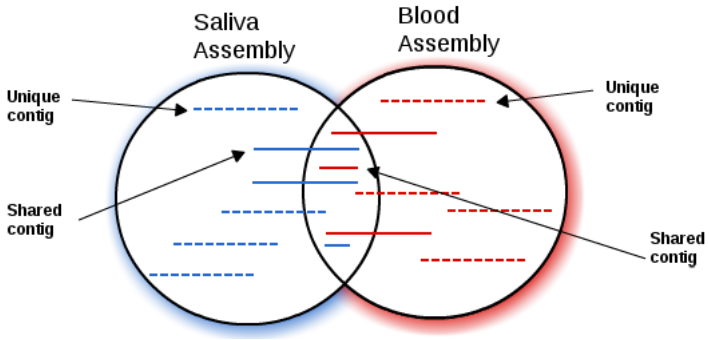

Figure 7: Blood and saliva assembly similarity. The DNA should be more or less the same on different tissues of particular organism, we wash out the unique contigs (has been found in only one of the tissue) to see high quality overlapped conigs.

Table 2: Assembly parameter permutations of one of the saliva sample

kmer	cvCut	exp	ctgs	asmLg	N50	mean	1k	max	tiles	rdPc
19	1	2	401882	22622920	72	56	0	768	1557292	13.79
19	10	20	1498	61831	71	41	0	719	42986	0.38
19	2	4	123076	7746290	89	62	1	1080	858101	7.60
19	3	6	36268	2120335	96	58	6	2027	340165	3.01
19	auto	auto	401952	22623670	72	56	0	768	1557200	13.79
21	1	2	171074	13779094	79	80	0	845	1073300	9.50
21	10	20	306	29951	180	97	2	1289	31834	0.28
21	2	4	53314	4924451	97	92	0	948	600471	5.32
21	3	6	11927	1153377	112	96	1	1169	220671	1.95
21	auto	auto	171079	13779194	79	80	0	845	1073300	9.50
23	1	2	104915	9085789	83	86	0	675	713364	6.32
23	10	20	137	20275	218	147	0	887	31586	0.28
23	2	4	33675	3342705	99	99	1	1189	400290	3.54
23	3	6	6559	702849	112	107	3	1189	134814	1.19
23	auto	auto	104917	9085833	83	86	0	675	713364	6.32
25	1	2	50843	4792063	91	94	0	809	428658	3.80
25	10	20	85	14970	223	176	0	644	13519	0.12
25	2	4	20510	2105780	101	102	1	1093	260509	2.31
25	3	6	3802	422759	111	111	1	1093	86326	0.76
25	auto	auto	50843	4792063	91	94	0	809	428658	3.80
27	1	2	27141	2678456	95	98	0	868	254768	2.26
27	10	20	61	11668	253	191	0	643	16682	0.15
27	2	4	11667	1236771	103	106	0	868	162892	1.44
27	3	6	2082	243648	114	117	0	868	59956	0.53
27	auto	auto	27141	2678456	95	98	0	868	254768	2.26
29	1	2	12390	1289795	100	104	0	662	132278	1.17
29	10	20	36	6781	229	188	0	613	5271	0.05
29	2	4	5953	656236	106	110	0	662	88727	0.79
29	3	6	1078	136116	122	126	0	677	33183	0.29
29	auto	auto	5956	656320	106	110	0	662	88727	0.79
31	1	2	5082	557770	105	109	0	677	66992	0.59
31	10	20	20	3692	210	184	0	420	3026	0.03
31	2	4	2729	315809	110	115	0	677	48924	0.43
31	3	6	563	77382	130	137	0	691	21924	0.19
31	auto	auto	2729	315809	110	115	0	677	48924	0.43

Table 3: Most frequent item on blood annotation

Protein Name	Frequency	Mean Identity	Mean Score bit	Mean reference length	Mean contigs length
ref—XP-002834723.1—	2201	45.08	40.41	339	89.30
ref—XP-002372134.1—	2060	44.23	39.76	477	85.26
ref—XP-001728577.1—	1511	45.87	40.18	336	90.70
ref—XP-001728578.1—	1380	45.21	39.55	285	86.91
ref—XP-002372114.1—	1109	46.52	40.09	312	90.66
ref—XP-002833573.1—	1051	45.44	40.37	450	89.07
gb—EAW55841.1—	939	51.92	41.14	528	103.12
ref—ZP-09921920.1—	869	44.28	38.74	228	79.26
ref—XP-002834930.1—	829	41.87	38.08	333	80.26
gb—AAO61995.1—	737	52.12	39.29	195	104.87
ref—XP-003641012.1—	681	39.34	37.81	579	72.59
ref—XP-002404893.1—	619	46.44	41.06	165	87.67
ref—XP-002833300.1—	584	43.79	39.32	273	88.46
gb—EGF47254.1—	474	44.99	39.34	213	86.36
gb—EAW62577.1—	471	41.21	38.33	2070	79.01
gb—EHB07202.1—	420	44.96	38.09	204	85.73
ref—ZP-04563055.1—	404	52.69	39.61	171	111.85
ref—XP-002833510.1—	384	39.92	37.75	279	75.88
gb—EFB17036.1—	369	42.68	39.33	177	83.76

Table 4: Most frequent item on blood and saliva annotation

Protein Name	Frequency	Mean Identity	Mean Score bit	Mean reference length	Mean contigs length
gb—EID31395.1—	11	187.10	187.36	122.10	6234
ref—YP-006309919.1—	11	180.27	182.18	117.82	6237
ref—ZP-07828254.1—	9	159.67	162	107.44	4191
gb—EGL76648.1—	9	142.67	162	99.11	4287
gb—EIG39935.1—	9	106.33	108	71.22	6789
ref—ZP-08148246.1—	8	121.5	123.375	83.75	3447
ref—ZP-07827124.1—	8	106.5	106.875	72.75	11493
gb—AFE64322.1—	8	105	130.125	72.375	2472
ref—ZP-04598604.1—	8	106.125	106.5	72.375	10593

Discussion

The first task in any assembly experiment is to find the best way of assembling the reads to get the highest ratio of usable information. When doing an assembly of unmapped reads that may not only contain unknown regions of the human genome but also potentially thousands of micro-organisms the step of finding the the most effective parameters become even more complex. When evaluating the two methods used in this project we have to consider both the N50 values and contig length.

In evaluating these parameters the kmer value is a product of the settings used and the parameters producing the best set of contigs are hard to find.Although choosing the high K-mer and coverage cutoff can lead us to the longer contigs by ruling out the short conigs which appear at less accurate parameter setting but this can reduce the length of assembly [14]. However when using less stringent settings the number of false positives would of course also be higher.

To make the assembly is better to have more reads to find more overlap but there is a positive relation between the amount of reads and memory usage and by merging all the reads from different samples and run assembly as in the first approach used (method 1) memory usage increase dramatically. To increase the assembly speed and reduce the memory usage it's better to prone the reads which do not have good quality and just make the assembly more complex and assemble each sample individually then merge the contig results as in the second approach used (method 2).

When comparing the saliva and blood samples it may be concluded that the highest number of contigs was found in the saliva sample. This may be explained by the fact that you can find a lot of bacteria in saliva than blood and it should be affected by the food and other bacteria which can be find in the air. Also as blood are more sterile than saliva this sample should contain a lower amount of bacterial DNA.

As blood are more sterile the contigs found if true positives are more probable to contain regions and even genes that are still not annotated as a part of the human genome. To this end it is important to add that some of the contigs found in the blood did have a close match to regions that are part of the genome to species closely related to us. However, these result needs to validated by further study.

When comparing the contigs found in blood and saliva the number of shared contigs are only a fraction of the total amount of contigs found in either of the samples. The result is not surprising as the number of micro-organisms in the mouth would make up a large part of the genetic material collected and would also imply that the systematical error are low. It is also highly possible that contigs found in both samples and that resembles ancestral DNA are previously unknown part of the human genome as they have been found independently of each other in two different samples.

It can also be concluded that a low level contamination have been introduced in the blood sample as one of the most frequent protein in the blood sample result belong to a bacteria which resides on the skin surface. This bacteria have probably been introduced to the sample in low levels during the blood extraction procedure showing the sensitivity of the method for finding bacteria present in low levels.

Of the contigs found only a small fraction was found in any database. As this may imply a high level of contigs being false positives, we can not disregard the notion that these contigs may derive from micro-organisms previously not described or sequenced. Opening up the exciting possibility of a large fraction of bacteria living very close to us that we currently know nothing at all about.

Bibliography

[1] Sabyasachi S. 2000. Human Genome: The Book of Life.IITK Archive 3.

[2] Martinez II A. 2002. "Information management and the biological warfare threat".Thesis, Naval Postgraduate School, United State Navy.

[3] Wright FA, Lemon WJ, Zhao WD, Sears R, Zhuo D, Wang JP, Yang HY, Baer T, Stredney D, Spitzner J, Stutz A, Krahe R, and Yuan B. 2001.A draft annotation and overview of the human genome. Genome Biology, 2:1-18

[4] Pennisi, E. 2003. A Low Number Wins the GeneSweep Pool. Science 300:1484.

[5] Li H. 2011. The human reference genome(Understanding GRCh37):http://lh3lh3.users.sourceforge.net/humanref.shtml. Date visited 27 June 2012

[6] Admin. 2009. About the Human Feb. 2009 (GRCh37/hg19) assembly - Assembly Details: http://genome-mirror.duhs.duke.edu/cgi-bin/hgGateway

[7] Bronwen G. 2011. Accessing alternate sequences in human: http://www.ensembl.info/blog/2011/05/20/accessing-non-reference-sequences-in-human. Date visited 27 June 2012.

[8] Wang J, Wang W, Li R, Li Y, Tian G, Goodman L, Fan W, Zhang J, Li J, Zhang J, Guo J, Feng B, Li H, Lu Y, Fang1 X, Liang H, Du ZH, Li D, Zhao Y, Hu Y, Yang ZH, Zheng HA, Hellmann I, Inouye M, Pool J, Yi1 X, Zhao J, Duan J, Zhou Y, Qin J, Ma LI, Li GU, Yang ZH, Zhang G, Yang B, Yu C, Liang F, Li W, Li Sh, Li D, Ni P, Ruan J, Li Q, Zhu Ho, Liu Do, Lu Zh, Li N, Guo G, Zhang Ji, Ye J, Fang L, Hao Q, Chen Q, Liang Y, Su Y, San A, Ping C,Yang S, Chen F, Li Li, Zhou K, Zheng H, Ren Y, Yang L, Gao Y,Yang G, Li Zh, Feng XI, Kristiansen K, Wong G, Nielsen R,Durbin R, Bolund L, Zhang X, Li S, Yang H & Wang J. 2008. The diploid genome sequence of an Asian individual. Nature 456, 60-65

[9] Li R,Zhu H, Ruan J, Qian Wu, Fang Xi, Shi Zho, Li Yi, Li S, Shan G, Kristiansen Ka,Li S,Yang H, Wang J and Wang J. 2010. De novo assembly of human genomes with massively parallel short read sequencing. Genome Research. 20, 265-272

[10] Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, Marth G, Abecasis G, Durbin R. 2009. The Sequence Alignment/Map format and SAMtools. Bioinformatics 25:2078-2079.

[11] Hohl M, Kurtz S, Ohlebush E. 2002. Efficient multiple genome alignment. Bioinformatics 18: 312-320.

[12] Samanta. 2011.De Bruijn graphs I :http://www.homolog.us/blogs/2011/07/28/de-bruijn-graphs-i.Date visited 27 June 2012

[13] Leipzig J. 2012. Velvet Assembly Report. http://code.google.com/p/standardized-velvet-assembly-report. Date visited 27 June 2012.

[14] Leipzig J. 2009. Using Vmatch to combine assemblies: http://jermdemo.blogspot.se/2009/11/using-vmatch-to-combine-assemblies.html. Date visited 27 June 2012

[15] Kleffe J, Moller F and Wittig B. 2007. Simultaneous identification of long similar substrings in large sets of sequences. BMC Bioinformatics 8:7

[16] Kleffe J, Moller F and Wittig B. 2006. ClustDB: A High-Performance Tool for Large Scale Sequence Matching. Database and Expert Systems Applications , doi 10.1109/DEXA.2006.40.

[17] Barsky M, Stege U, Thomo A, and Upton C. 2009. Suffix tree for very large genomic sequences. ACM CIKM, pages 1417-1420.

[18] Ewing B, Hillier L, Wendl MC, Green P. 1998. Base-calling of automated sequencer traces using phred. I. Accuracy assessment. Genome Resource 8: 175 - 185

[19] Cock P, Fields C, Goto N, Heuer M, and Rice P. 2010. The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. Nucl. Acids Res 38: 1767-1771.

Supplementary material

Source code for various scripts used in the project

Script 1: Preparing data

```
#! /bin/bash -l
############################
## Writen by AMIN SAFFARI
## Preparing data set for velvet assembler
############################
fileName=$1
Datapath=$2
echo "++++++++++++++++++"
echo "Preparing data start on" $(date)
echo "Sam file PATH is: "
for f in 'ls ${Datapath}/*.sam'
do
  cd $Datapath
  echo "--> $f"
  echo "    |Splite F3 R3 .csfasta"
  awk '{if (($14 !~/[.]/) && (substr($1,1,1)!="@")) {if (substr($14,6,1)=="T"){print ">"$1"_F3","\n"substr($14,6,length($14)) > substr(FILENAME,1,length(FILENAME)-4)"_F3.csfasta" } else {print ">"$1"_R3","\n"substr($14,6,length($14)) > substr(FILENAME,1,length(FILENAME)-4)"_R3.csfasta" }}}' $f
  echo "    |Splite F3 R3 .qual"
  awk '{if (($14 !~/[.]/) && (substr($1,1,1)!="@")) {if (substr($14,6,1)=="T"){print ">"$1"_F3","\n"substr($13,6,length($13)) > substr(FILENAME,1,length(FILENAME)-4)"_F3_QV_Phred.qual" } else {print ">"$1"_R3","\n"substr($13,6,length($13)) > substr(FILENAME,1,length(FILENAME)-4)"_R3_QV_Phred.qual" }}}' $f
  echo " Convert it to ASCII value"
  perl ~/Package/Phred2csquality.pl $f
  echo "Ouput directory is /proj/b2010007/private/twins/concatenated"
  echo "Concat fileName is ${fileName}"
  echo "++++++++++++++++++"
  echo "Sorting"
  time radsort ${fileName}_F3.csfasta 5 > temp.${fileName}_F3.csfasta
  rm ${fileName}_F3.csfasta
  mv temp.${fileName}_F3.csfasta ${fileName}_F3.csfasta
  echo "Sorting ${fileName}_F3.csfasta is finished"
  time radsort ${fileName}_R3.csfasta 5 > temp.${fileName}_R3.csfasta
  rm ${fileName}_R3.csfasta
  mv temp.${fileName}_R3.csfasta ${fileName}_R3.csfasta
  echo "Sorting ${fileName}_R3.csfasta is finished"
  time radsort ${fileName}_F3_QV.qual 5 > temp.${fileName}_F3_QV.qual
  rm ${fileName}_F3_QV.qual
  mv temp.${fileName}_F3_QV.qual ${fileName}_F3_QV.qual
  echo "Sorting ${fileName}_F3_QV.qual is finished"
  time radsort ${fileName}_R3_QV.qual 5 > temp.${fileName}_R3_QV.qual
  rm ${fileName}_R3_QV.qual
  mv temp.${fileName}_R3_QV.qual ${fileName}_R3_QV.qual
  echo "Sorting ${fileName}_R3_QV.qual is finished"
  echo "Run preprocessing"
  time perl ~/Package/SOLiD_preprocess/SOLiD_preprocess_filter_v2.pl -i mp -f ${fileName}_F3.csfasta -g ${fileName}_F3_QV.qual -r ${fileName_R3.csfasta -s ${fileName}_R3_QV.qual -p y -q 24 -n y -a y -o ${fileName}_filter_ouput -v n
  echo "Completed on" $(date)
  echo "----------------------"
done
```

Script 2: Guess encoding

```
############################
## Modiffied by AMIN SAFFARI
## Find the encoding range
############################

import sys
import optparse
RANGES = {
    'It is Sanger': (33, 73),
    'It is Solexa': (59, 104),
    'It is Illumina-1.3': (64, 104),
    'It is Illumina-1.5': (67, 104)
}
def find_range(q_str):
    Dig = [ord(c) for c in q_str]
    return min(Dig), max(Dig)

def Encodings_in_range(RMin, RMax, rang=RANGES):
    valid_enc = []
    for encod, (emin, emax) in rang.items():
        if RMin >= emin and RMax <= emax:
            valid_enc.append(encod)
    return valid_enc

def main():
    pARG = optparse.OptionParser(__doc__)
    opts, args = pARG.parse_args()
    print >>sys.stderr, "# reading qualities"
    G_Min, G_Max  = 99, 0
    Array = []
    for Z, line in enumerate(sys.stdin):
        L_Min, L_Max = find_range(line.rstrip())
        if L_Min < G_Min or L_Max > G_Max:
            GMin, GMax = min(L_Min, G_Min), max(L_Max, G_Max)
            Array = Encodings_in_range(G_Min, G_Max)
            if len(Array) == 0:
                print >>sys.stderr, "There is no any encodings on range: %s" % str((G_Min, ↩
                    G_Max))
                sys.exit()
            if len(Array) == 1 and opts.n == -1:
                print "\t".join(Array) + "\t" + str((G_Min, G_Max))
                sys.exit()
        if opts.n > 0 and Z > opts.n:
            print "\t".join(Array) + "\t" + str((G_Min, G_Max))
            sys.exit()
    print "\t".join(Array) + "\t" + str((G_Min, G_Max))


if __name__ == "__main__":
    import doctest
    if doctest.testmod(optionflags=doctest.ELLIPSIS |\
                                   doctest.NORMALIZE_WHITESPACE).failed == 0:
        main()
```

Script 3: Convert phred to csquality

```
#!/usr/bin/perl
############################
## Writen by AMIN SAFFARI
## convert Phred quality to csQuality format
############################
#the filename should be passed as a parameter so #ARGV+1 should not be 0
if(!($#ARGV+1)){
print ("the filename should be passed in as a parameter\n");
exit;
}
$inputFile=$ARGV[0];
$outputFile=substr($inputFile,0,length($inputFile)-11).'.qual';
open (IN_FILE, "<$inputFile") || die "Could not read from $inputFile, program halting.\n$!";
open(OUT_FILE,">>$outputFile") || die "Could not write to $outputFile, Program halting.\n$!"
    ;
chomp(my $record = <IN_FILE>);
while($record ){
    #Check the line which is contain "R3 or F3"
    if(($record =~/_R3/) || ($record =~/_F3/)) {
      print OUT_FILE "$record"."\n";
    } else  {
      @ascii_character_numbers = unpack("C*", "$record");
      $size=scalar(@ascii_character_numbers);
      for($i = 0; $i < $size; ++$i) {
        $ascii_character_numbers[$i] = $ascii_character_numbers[$i] -33;
      }
      print OUT_FILE "@ascii_character_numbers"."\n";
    }
    chomp($record = <IN_FILE>);

}
close(IN_FILE);
close(OUT_FILE);
print("Converte $inputFile to $outputFile is finished \n");
exit;
```

Script 4: Assembly pipeline

```
#! /bin/bash -l
############################
## Modified by AMIN SAFFARI
## Assembly pipline
############################
ulimit -s unlimited
ulimit -v 240000000
now=$1
cd $2
for seqDir in  `ls /proj/unmapped_reads/Data/de_mates__input.de | cut -d'.' -f 1`
do
    datapath=`dirname $seqDir`
    filename=`basename $seqDir`
    outdir=`echo $datapath | cut -d'/' -f 6`
    echo "**************************"
    echo "Assembling $outdir start"
    echo "++++++++++++++++++++++++++"
    echo "datapath is $datapath"
    echo "filename is $filename"
    echo "outdir is $now/$outdir"
    echo "++++++++++++++++++++++++++"
    if [ -z "$seqDir" ]
    then
        echo "Pass sequence file without the extension"
        exit
    fi
    cd $now
    ext=$(ls $seqDir* | grep -oE -m1 '(.fa|.de|.fasta|.fastq|.eland|.gerald)?$')
    if [ ! -e "$seqDir""$ext" ]
    then
        echo "Where is file Cannot find $seqDir""$ext"
        exit
    fi
    if [ ! -e $outdir ]
    then
        mkdir $outdir
    fi
    cd $outdir
    longExt="`echo $ext | sed -e 's/\.fa\(sta\)*/fasta/;s/\.de/fasta/;s/^\.//;'`"
    echo $longExt
    if [ ! -e $filename".stat" ]
    then
        echo "Cannot find $filename.stat. Starting to run fastaAllSize $seqDir$ext > ↩
            $filename.stat on your sequences."
        perl ~/Package/velvetReport/fastaAllSize $seqDir$ext > $filename".stat"
    fi
    for kmer in 31 29 27 25 23 21 19
    do
        if [ -e "out_"$kmer"/Roadmaps" ]
        then
        echo "roadmap ready"
        else
        echo "running velveth"
        ~/bin/velveth_de "out_"$kmer $kmer -$longExt -shortPaired $seqDir$ext
        fi
        if [ ! -e "out_"$kmer"/Roadmaps" ]
        then
        echo "velveth did not run normally on $outdir kmer=$kmer"
        exit
        fi
        for cvCut in 0 1 2 3 10
        do
            expCov=$((2*$cvCut))
        echo $expCov
        if [ "$expCov" -ne 0 ]
        then
           dirName="out_"$kmer"_"$cvCut"_"$expCov"_dir"
        else
           dirName="out_"$kmer"_auto_auto_dir"
        fi
        if [ -e $dirName"/contigs.fa" ] || [ -e $dirName"/contigs.fa.gz" ]
        then
            echo "I see contigs in $dirName"
        else
            mkdir $dirName
```

```
            ln -s $seqDir$ext $dirName"/reads"$ext
            ln -s "../"$filename".stat" $dirName"/reads.stat"
            ln -s "../out_"$kmer/Sequences $dirName"/Sequences"
            ln -s "../out_"$kmer/Roadmaps $dirName"/Roadmaps"
            cp "out_"$kmer/Log $dirName"/Log"
            echo  $dirName
            echo velvetg
                if [ "$expCov" -ne 0 ]
                then
                      ~/bin/velvetg_de $dirName -exp_cov $expCov -cov_cutoff $cvCut -↩
                          read_trkg yes -amos_file yes -unused_reads yes
                else
                  ~/bin/velvetg_de $dirName -exp_cov auto -cov_cutoff auto -read_trkg yes -↩
                      amos_file yes
            fi
            if [ $? != 0 ]
            then
            echo "velvetg did not run normally"
            echo "$?"
                exit
            else
            echo "velvetg finished"
            fi
        fi
         if [ -e $dirName"/contigs.txt" ]
              then
                 echo "I see contigs.txt in ${dirName}, skip postProcessing "
                 if [ -e $dirName"/contigs.out" ] && [ "`wc -l $dirName"/contigs.out" | ↩
                     cut -f1 -d' ' `" -ge 10 ]
                 then
                     echo "I see some lines in ${dirName}/contigs.out, skip denovoadp "
                     if [ -e $dirName"/metadata.txt" ]
                     then
                        echo "I see metadata.txt in ${dirName}"
                     else
                         echo "Plan to create metadata.txt"
                         perl ~/Package/velvetReport/generateAssemblyStats.pl -useamos ↩
                             $dirName > $dirName"/metadata.txt"
                     fi
                 else
                     rm $dirName"/contigs.out"
                     echo "Plan to create contigs.out"
                     ~/bin/denovoadp $dirName/contigs.txt 10000 > $dirName/contigs.out
                     if [ -e $dirName"/contigs.out" ] && [ "`wc -l $dirName"/contigs.out"↩
                          | cut -f1 -d' ' `" -ge 10 ]
                     then
                rm $dirName"/metadata.txt"
                                 echo "Plan to create metadata.txt"
                                 perl ~/Package/velvetReport/generateAssemblyStats.pl -↩
                                     useamos $dirName > $dirName"/metadata.txt"
                     else
                         rm $dirName"/contigs.out"
                         rm $dirName"/contigs.txt"
                         echo "Some thing was wrong in ${dirName}/contigs.txt so I remove↩
                              it for you"
                         #exit
                     fi
                 fi
                 else
                     echo "start postProcessing"
                     perl ~/Package/copy_denovo_postprocessor_solid_v1.6.pl --afgfile ↩
                         $dirName"/velvet_asm.afg" --csfasta $datapath"/cs_mates_input.↩
                         csfasta"  --output $dirName/contigs.txt
                     echo "Plan to create contigs.out"
                     ~/bin/denovoadp $dirName/contigs.txt 10000 > $dirName/contigs.out
                     echo "Plan to create metadata.txt"
                     perl ~/Package/velvetReport/generateAssemblyStats.pl -useamos $dirName >↩
                          $dirName"/metadata.txt"
                 fi
    done
    done
echo "Assembling $outdir finished"
echo "**************************"
echo "------------------------------"
done
```