universität wien

DISSERTATION

Titel der Dissertation

„Benchtop sequencing on benchtop computers“

Verfasser

DI (FH) Fritz Joachim Sedlazeck

angestrebter akademischer Grad

Doktor der Naturwissenschaften (Dr.rer.nat.)

Wien, 2012

Studienkennzahl lt. Studienblatt:	A 091 490
Dissertationsgebiet lt. Studienblatt:	Molekulare Biologie
Betreuerin / Betreuer:	Univ.- Prof. Arndt von Haeseler

An expert is a man who has made all the mistakes which can be made, in a narrow field.

Niels Bohr (1885-1962))

If you can't explain it simply, you don't understand it well enough.

Albert Einstein (1879-1955)

Abstract

Next Generation Sequencing (NGS) is a powerful tool to gain new insights in molecular biology. With the introduction of the first bench top NGS sequencing machines (e.g. Ion Torrent, MiSeq), this technology became even more versatile in its applications and the amount of data that are produced in a short time is ever increasing. The demand for new and more efficient sequence analysis tools increases at the same rate as the throughput of sequencing technologies. New methods and algorithms not only need to be more efficient but also need to account for a higher genetic variability between the sequenced and annotated data. To obtain reliable results, information about errors and limitations of NGS technologies should also be investigated. Furthermore, methods need to be able to cope with contamination in the data.

In this thesis we present methods and algorithms for NGS analysis. Firstly, we present a fast and precise method to align NGS reads to a reference genome. This method, called NextGenMap, was designed to work with data from Illumina, 454 and Ion Torrent technologies, and is easily extendable to new upcoming technologies. We use a pairwise sequence alignment in combination with an exact match filter approach to maximize the number of correctly mapped reads. To reduce runtime (mapping a 16x coverage human genome data set within hours) we developed an optimized banded pairwise alignment algorithm for NGS data. We implemented this algorithm using high performance programing interfaces for central processing units using SSE (Streaming SIMD Extensions) and OpenCL as well as for graphic processing units using OpenCL and CUDA. Thus, NextGenMap can make maximal use of all existing hardware no matter whether it is a high end compute cluster or a standard desktop computer or even a laptop. We demonstrated the advantages of NextGenMap based on real and simulated data over other mapping methods and showed that NextGenMap outperforms current methods with respect to the number of correctly mapped reads.

The second part of the thesis is an analysis of limitations and errors of Ion Torrent and MiSeq. Sequencing errors were defined as the percentage of mismatches, insertion and

deletions per position given a semi-global alignment mapping between read and reference sequence. We measured a mean error rate for MiSeq of 0.8% and for Ion Torrent of 1.5%. Moreover we identified for both technologies a non-uniform distribution of errors and even more severe of the corresponding nucleotide frequencies given a difference in the alignment. This is an important result since it reveals that some differences (e.g. mismatches) are more likely to occur than others and thus lead to a biased analysis. When looking at the distribution of the reads accross the sample carrier of the sequencing machine we discovered a clustering of reads that have a high difference ($> 30\%$) compared to the reference sequence. This is unexpected since reads with a high difference are believed to origin either from contamination or errors in the library preparation, and should therefore be uniformly distributed on the sample carrier of the sequencing machine.

Finally, we present a method called DeFenSe (Detection of Falsely Aligned Sequences) to detect and reduce contamination in NGS data. DeFenSe computes a pairwise alignment score threshold based on the alignment of randomly sampled reads to the reference genome. This threshold is then used to filter the mapped reads. It was applied in combination with two widely used mapping programs to real data resulting in a reduction of contamination of up to 99.8%. In contrast to previous methods DeFenSe works independently of the number of differences between the reference and the targeted genome. Moreover, DeFenSe neither relies on ad hoc decisions like identity threshold or mapping quality thresholds nor does it require prior knowledge of the sequenced organism.

The combination of these methods may lead to the possibility of transferring knowledge from model organisms to non model organisms by the usage of NGS. In addition, it enables to study biological mechanisms even in high polymorphic regions.

Parts of this thesis have been published in the following articles:

1. P. Rescheneder, A. von Haeseler, and F.J. Sedlazeck (2011) MASon: Million Alignments In Seconds - A Platform Independent Pairwise Sequence Alignment Library for Next Generation Sequencing Data. Proceedings of the International Conference on Bioinformatics Models, Methods and Algorithms (BIOINFORMATICS 2012), 195-201, SciTePress, Setubal, Portugal. (DOI: 10.5220/0003775701950201)

All methods including developments presented in this thesis are freely available from `http://www.cibiv.at/software/ngm`.

Acknowledgments

I want to thank my supervisor Arndt von Haeseler who gave me the opportunity and the autonomy to choose and to work on a topic that I was interested in. In addition, I want to thank Ingo Ebersberger who provided valuable feedback throughout the process of writing this thesis.

Moreover, I would like to say a big thank you to my colleges, especially Philipp Rescheneder for his helpful feedback and fruitful collaborations.

Finally, I owe sincere and earnest thankfulness to my girl friend Barbara and my parents Gertrude und Josef for their constant support over the past years.

Contents

List of Figures

List of Tables

Chapter 1

Overview

1.1 Motivation

The advent of Next Generation Sequencing (NGS) has led to new insights in molecular biology. Current emerging technologies like benchtop sequencing are boosting NGS to become a standard method in molecular biology (Glenn, 2011). The wide spread usage of NGS technologies requires fast and inexpensive ways to analyse the data. In recent years the usage of clusters and cloud computing systems where discussed (Schatz, 2009; Niemenmaa *et al.*, 2012; David *et al.*, 2011). However, what if one has no access to clusters or the conditions of the project prohibit the distribution of the data (e.g. personalized data)? A cheap and easily accessible solution would be the analysis on desktop computers. Although they are less powerful in terms of numbers of computation per second or amount of memory, it should be possible to adopt technologies that the gaming industry has develop to speed up computations. These could be then used either to reduce the runtime of sequence analysis tools or to perform more computations, e.g. to analyse reads with a higher number of differences compared to the reference. Since methods like Bowtie (Langmead *et al.*, 2009) or BWA (Li and Durbin, 2009) already achieve a short runtime one would focus on the latter explicitly, i.e. more throughout searches. Note in this context that, new NGS technologies like Ion Torrent have the trend to produce longer reads with a higher number of errors. Thus, more throughout search methods are inevitable to successfully align those reads to a reference sequence (Glenn, 2011).

To obtain reliable results after the initial processing of NGS data one has to account

for systematic errors or biases introduced by the experimental setting (e.g. batch effect, preparation error, sequencing error) or the used software. Typically one expects the software to be well tested and to return the correct results. This is often demonstrated by several benchmarks done by the developers. In contrast to the used software it is often not trivial to assess the effect on the experimental results of used sequencing technologies and experimental protocols. The batch effect and effects of DNA amplification are well known and method exist to account for this (e.g. Li and Rabinovic (2007)). However, the influence of the used NGS technology is difficult to estimate. A few papers were published to measure the sequencing error for NGS technologies (Huse *et al.*, 2007; Suzuki *et al.*, 2011; Glenn, 2011; Loman *et al.*, 2012), still it is often unclear what errors or biases to expect. Possible biases in the frequency of inserted nucleotides will influence latter analysis. Furthermore, this might not only be related to the sequencing technology but further might differ from machine to machine.

Another cause of error is contamination introduced during or before sample preparation. Here, we define contamination as reads that origin from a different organism than intended to be sequenced. Especially in cases like ChIP or RNA sequencing one relies on a clustering of reads at a specific region on the genome. The conclusions drawn from such an analysis would be severely compromised if part of the reads came from a different genomic region or even from a different organism/species than sequenced. Detection of contamination is not trivial since the percentages of mismatches, insertions and deletions for a read when aligned to the reference depends on the sequencing technology but also on the genetic diversity of a locus (e.g. how many differences can we expect between the organism and the annotated genome). It would be ideal to have a method that automatically adjusts to the underlying experimental settings and detect reads that should not be taken into account.

Currently it is not foreseeable how the advent of NGS technologies will change our viewpoint of genetics. It is clear that NGS already has a deep impact in the way science is currently done (Zhang *et al.*, 2011). To gain further insights based on NGS technologies several issues have to be addressed. Those mentioned before represent only the tip of the iceberg but might open the field for other experiments. One example would be the mapping of sequenced genomes of a population to a distantly related annotated reference genome. This may lead to a fast and cheap way to identify population specific mechanisms like rearrangements, inversion, recombination hotspots, the estimation of

site frequency spectra of SNPs etc. .

1.2 Contributions

We have addressed a number of issues in the field of Next Generation Sequence analysis. To this end, a number of algorithms have been developed and implemented. The results of this work are presented in the subsequent chapters.

Chapter 2 gives a brief introduction to NGS. Frequently used sequencing technologies are briefly described and discussed. Furthermore, it provides a brief introduction in reference based sequence mapping and de novo assembly of NGS data.

Chapter 3 presents an efficient mapping approach (NextGenMap) utilizing modern hardware resources. The algorithm focuses on correctly mapping a high number of reads. Real and simulated data analysis show the improvement over other mapping methods.

Chapter 4 presents an optimized algorithm to compute pairwise alignments for NGS data . The algorithm was implemented with a variety of different high performance APIs (SSE, OpenCL, Cuda) and is designed to utilize current CPUs as well as graphic cards. This algorithm is used in the program of NextGenMap and DeFenSe (see chapters 3 and 6).

Chapter 5 presents an error rate analysis of Ion Torrent and MiSeq data. It outlines advantages and disadvantages of each technology and gives insights about platform specific sequencing errors.

Chapter 6 presents DeFenSe a method to automatically detect and reduce contamination in NGS data. It uses the algorithm from chapter 4 to compute an alignment score threshold based on the experimental data.

Chapter 7 gives a brief summary of the results obtained in this thesis.

Chapter 2

Next Generation Sequencing

2.1 Next Generation Sequencing Technologies

2.1.1 Introduction

The Human Genome Project (The International Human Genome Sequencing Consortium, 2001) started in 1989 with the goal to reveal the human genome information by the use of DNA sequencing. This is the process of determining the order of the nucleotides (adenine, guanine, cytosine and thymine) along a DNA molecule. In 2001, after 12 years, a human draft genome was published (The International Human Genome Sequencing Consortium, 2001). Now 11 years later sequencing on a genome level has become a routine method in molecular biology.

Genome-sequencing leads to insights about the gene structure and genotype variation within a population, and to the discovery of genetic causes of diseases. Previous strategies like Sanger sequencing (Sanger *et al.*, 1977), and even more so chemical sequencing following Maxam−Gilbert (Maxam and Gilbert, 1977), were cost and labor intensive. Furthermore, they did not provide a large amount of sequence information. This led to the development of parallel capillary sequencing machines. Still the amount of sequence information was not efficient to study genomes of various organisms.

Next Generation Sequencing (NGS) technologies such as 454 and Illumina significantly increased the amount of sequence information compared to previous technologies (MacLean *et al.*, 2009; Glenn, 2011). At the same time NGS reduced the costs for experiments for sequencing a human genome from several million down to few thousand

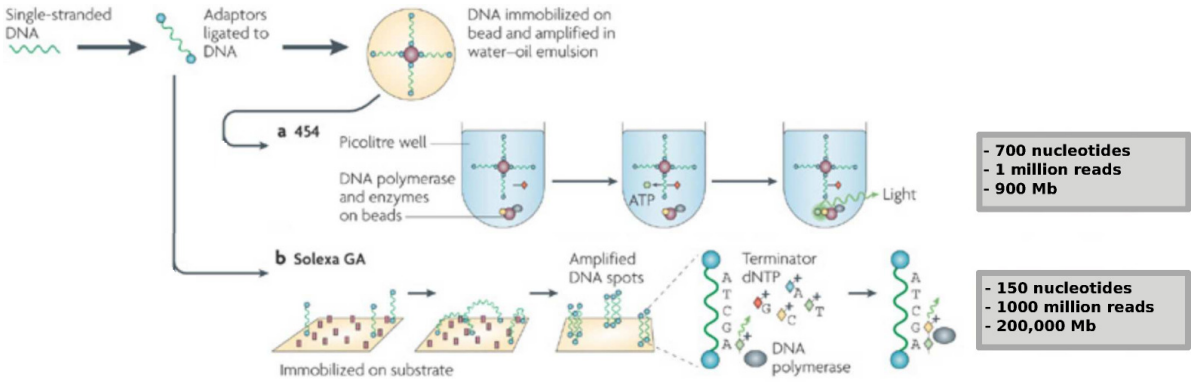

Figure 2.1: The principle of sequencing technologies. Image taken from Nature Reviews MacLean *et al.* (2009) and modified according to (Glenn, 2011)

dollars. Another difference between NGS and the former used sequencing technologies is that NGS uses a real time sequencing technology. This means during the PCR reaction the machine retrieves information which nucleotide was incorporated. In contrast to e.g. Sanger sequencing, where the included nucleotide is determined long time after the PCR reaction took place.

There are various applications where NGS is applied to. Apart from Genome-sequencing it is used in RNA sequencing, which gives deeper insights about the genes and their expression levels. ChIP- sequencing studies are used to identify genomic regions with interesting features (depending on the experimental goal), for example transcription factor binding sites.

Figure 2.1 summarizes the principles of 454 and Illumina (Solexa) sequencing technologies. The detailed technological aspects of each sequencing technologies have been extensively described and discussed elsewhere (Hall, 2007; Holt and Jones, 2008; Medini *et al.*, 2008; Shendure and Ji, 2008), therefore we only give a brief summary of technologies that are used in this thesis.

2.1.2 454 sequencing

454 was the first NGS technology on the market (Margulies *et al.*, 2005). It is based on pyro-sequencing, this is the process of determining the sequence by the intensity of emitted light signals. Figure 2.1 (a) displays the principle of the sequencing procedure.

The first step is the ligation of the adapter sequence to the sequence of interest. One of the adapters allows the sequence to bind on the library beads. After the binding PCR is used to amplify the template sequences on the individual beads. The sequencing process is then carried out on a PicoTiterPlate, which consist of approximately 1.6 million wells. Per well, a bead containing the sequence, a DNA polymerase and a chemiluminescent enzyme is placed. The sequencing process consists of different cycles. Per cycle only one type of dNTP (Desoxyribonukleosidtriphosphate, N= A, C, G, T) is added. If the DNA polymerase can incorporate this nucleotide, pyrophosphate is released and the nucleotide is attached to the sequence. If more than one nucleotide of the same type occurs as a substring, then all complementary nucleotides are incorporated within the same sequencing cycle. The pyrophosphate is used as a substrate for the Enzymatic reaction chain that leads to the emission of light intensity corresponding to the amount of pyrophosphate. Given the number of pyrophosphates released per cycle and the resulting intensity of the light, it is possible to determine the number of nucleotides that were incorporated. Per cycle an image is recorded that gives information about the light intensity and the read group where a nucleotide was attached. After each cycle, a washing step removes unused dNTPs and the next cycle starts with a different dNTP. The current read length limitation is $\sim$ 700 bp (Glenn, 2011).

2.1.3 Solexa / Illumina sequencing

The Illumina sequencing technology was launched in 2006. Figure 2.1 (b) displays the principle of the sequencing procedure. Similar to Sanger sequencing and 454, the technology is based on sequencing by synthesis (Mardis, 2008). In contrast to 454, where the sequencing reaction takes place on a bead, sequencing is performed on a glass slide. After the ligation of the adapter sequence, the template sequences are attached to a glass layer. In the next step, a PCR amplification takes place. The amplification step is called bridge amplification since the $3'$ and $5'$ ends of the template sequence are attached to the sequencing surface during the synthesis of the second strand. The result of the amplification are clusters with a large number of copies of the same sequence. Prior to the sequencing step, the reverse strand of the template sequence is cleaved and washed away. A DNA polymerase is used to incorporated ddNTP (nucleotides with reversible dye-terminators) to the sequence. After each addition of a ddNTP, the polymerase stops and the unincorporated ddNTP are washed out. The fluorophor of the added nucleotide

is excitated and an image of the flow cell is taken. Subsequently, the fluorophor and the terminators are cleaved, washed out and the next cycle begins.

The maximum read length of Illumina sequencing is currently 150 base pair. The technology is capable of producing paired end reads, i.e. both ends of a template can be sequenced. Thus, Illumina can determine up to 300 nucleotides per template.

2.1.4 Ion Torrent sequencing

The Ion Torrent sequencing technology was the first benchtop technology released 2011 from Life Technologies (Rothberg *et al.*, 2011). The sequencing technology is related to the 454 technology (Glenn, 2011). It uses beads where the fragmented DNA is bound to by one of the adapter sequences. Subsequent to PCR amplification of the bound template, each bead is loaded into individual sensor wells. All four nucleotides are added sequentially (dNTP). In contrast to 454, the insertion of a nucleotide is measured by the release of a proton during extension of the sugar-phosphate backbone of the newly synthesized DNA strand (Rothberg *et al.*, 2011). For each attached nucleotide a proton (H^+) is released and the pH change is detected by the sensor on the bottom of the well. A washing step to remove unused nucleotides and to readjust the pH is performed before a new cycle starts. Figure 2.2 displays the principle of the Ion Torrent technology. The current read length is up to ∼300bp.

2.1.5 MiSeq sequencing

The MiSeq system was the second benchtop sequencing technology released in 2011 by Illumina (Glenn, 2011). MiSeq uses the same technology as the Illumina technologies. Only the sequencing slide is smaller.

2.1.6 Discussion

The various Next Generation Sequencing (NGS) technologies open a broad field of biological questions that can be addressed with the help of DNA sequencing. This ranges from the determination of gene and genome sequences to the measurement of gene expression for RNA sequencing. However, sometimes one technology is more suited to

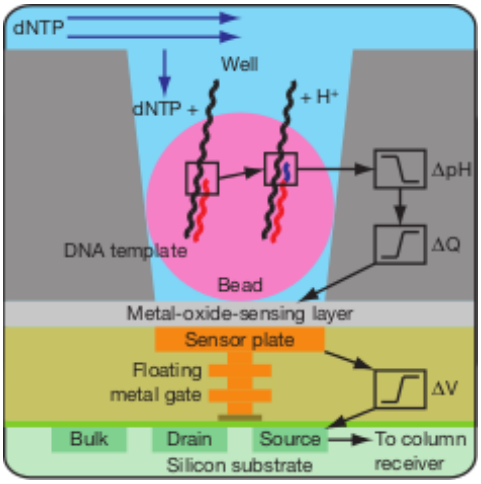

Figure 2.2: The general method of the Ion Torrent sequencing technology. Image taken from Rothberg *et al.* (2011).

answer a particular biological questions than the other. For example, 454 produces typically long reads, which allows the positioning of such on a reference sequence with higher confidence. On the other hand, 454 is known for having problems in determining the number of nucleotides incorporated per cycle (Huse *et al.*, 2007; Suzuki *et al.*, 2011; Glenn, 2011). The resulting increased probability for insertions and deletions (indels) can cause problems in the alignment that might not be encountered with Illumina data, where one expects to see mismatches as the main source of errors. (Suzuki *et al.*, 2011; Glenn, 2011).

Current technologies like benchtop sequencers (e.g. Ion Torrent) or single molecule sequencers (e.g. Pac Bio) promotes the wide use of sequencing technologies even further. However, their impact for molecular biology and medicine is yet not assessable.

Given the increased production of data and the reduction in costs it became a routinely used method in molecular biology. Nonetheless, it is important to note that they are not error free. Current NGS technologies show two different types of platform specific errors (either mismatches or indels). In principal this could be used in complementary sequencing approaches when correcting the errors of one technology by the other used technology and vice versa. However, in real life this is rarely the cases since only very few labs have access to two different sequencing technologies.

2.2 Computational approaches to analyze Next Generation Sequencing data

One of the first steps after obtaining the reads in every NGS analysis is either a reference based mapping (Trapnell and Salzberg, 2009), or a de novo assembly (Chaisson *et al.*, 2009) of the sequencing data.

Reference based mapping requires a genome of the same or at least a closely related species. The objective is to identify for a read the highest similar regions on the reference sequence. This has to be done for every read. Reference based mapping is used for various applications of NGS experiments, e.g. whole genome, RNA, or ChIP sequencing.

De novo assembly aims to identify reads with a highly similar overlap, which are then grouped together in contigs. The objective is to assemble the genome or at least a large fraction of it independent of a reference genome. One application of NGS experiments that de novo assembly can be used for is RNA sequencing, e.g. identification of novel splice variants, and whole genome sequencing.

When designing and implementing computational methods for NGS analysis one has to keep in mind the amount of data produced per sequencing run, and that the data maybe produced with different error characteristics (Flicek and Birney, 2009; Glenn, 2011). There are mainly two types of data; one consisting of long (>300bp) but few reads (e.g. 454 and Ion Torrent). And the other consisting of short reads ($\leq$<150bp) but millions of them (e.g. Illumina). These different characteristics of the two data types make it often hard or nearly impossible to have one algorithm for their analysis.

Currently there are many reference based mappers and de novo assemblers available that either focus on speed or on a high number of mapped reads (Li and Homer, 2010). Those that have speed as their main objective are based on Burrows Wheeler Transformation, e.g. Bowtie (Langmead *et al.*, 2009) and BWA (Li and Durbin, 2009). In contrast, applications that focus on a high number of mapped reads, e.g. SHRiMP2 (David *et al.*, 2011) and SSaha2 (Ning *et al.*, 2001), are based on pairwise sequence alignment algorithms.

Generally the choice of the mapping or assembly algorithm is strongly influenced by the experimental setup. For example, sequencing reads from the unknown genome sequence

of a species have to be processed by a de novo assembler. ChIP-Seq experiments, on the other hand, cannot be analyzed without a reference sequence or genome.

2.2.1 Reference based mapping

The main objective of a reference mapping program is to identify the region of the genome where a read originated from. Unlike earlier programs like Blast (Altschul *et al.*, 1990) or Blat (Kent, 2002), NGS mappers are designed to align short sequences to a closely related genome. The main assumption of every mapper is that there are just very few, if any, differences between the read and the corresponding genomic region. The number of maximally allowed differences is different for nearly every approach. Although to assume only a few, if any, differences seems to be a trivial assumption it is important for the design of mapping programs.

Figure 2.3 displays the general principle of contemporary methods. The Burrows-Wheeler as well as the seed based methods report for each read the starting position of the region in the reference with the lowest number of differences between the read and the region on the plus strand. This is typically reported either in the current standard formats SAM/BAM (Li *et al.*, 2009) or in their program specific formats. For a recent overview see (Schbath *et al.*, 2012).

The most frequently used reference based mapping programs, e.g. Bowtie (Langmead *et al.*, 2009) and BWA (Li and Durbin, 2009), are based on a suffix array that allows a sequence based search with up to n mismatches, insertions or deletions. The Burrows Wheeler Transformation (BWT) (Burrows and Wheeler, 1994) leads to very efficient running time. BWT is a lossles compression, which basically permutes and transforms the order of the sequence (Burrows and Wheeler, 1994). The longer the sequence the more one profits from the transformation. The BWT groups subsequences together that occur multiple times in the genome. This allows a fast access of subsequences. For example, using BWA, if there exists an exact matching read, BWA can identify its location in $O(m)$, where m is the read length. When searching for non exact matches BWA uses an approach to minimize the search space. The maximal number of differences allowed between a read and a region on the reference depends on the read length, e.g. it can be set to 2 differences for a 36bp read. High numbers of difference are possible, however they increase the runtime. Both methods BWA and Bowtie are designed to

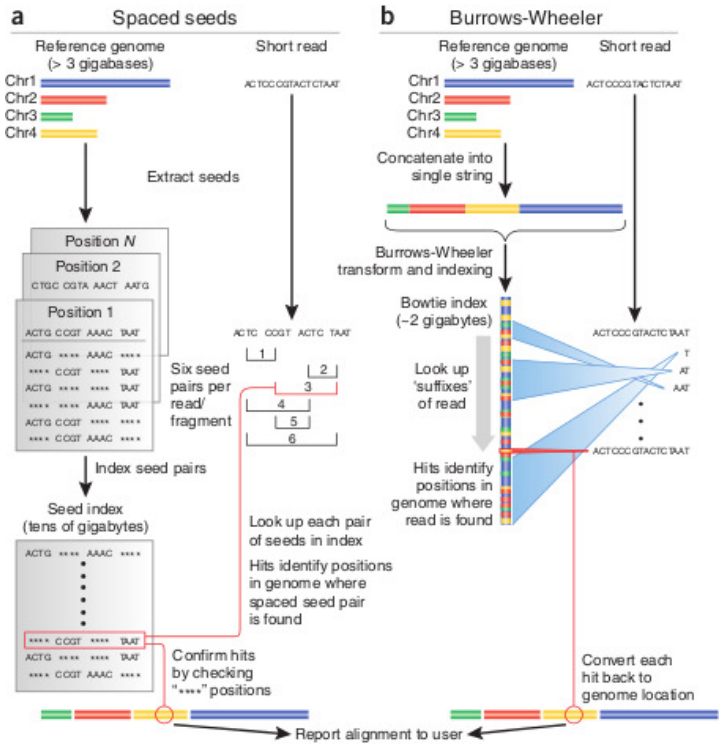

Figure 2.3: Generalized workflow of available mapping strategies. (a) describes the mechanism of a seed word approach. (b) displays the principle of Burrows Wheeler based mapping methods. Image taken from Trapnell and Salzberg (2009)

align a set of short sequences. Whereas BWA-SW is optimized to align long sequences (>100bp). Details of those methods have been described and discussed elsewhere (Flicek and Birney, 2009; Li and Durbin, 2009).

The focus of the second approach lies on a fast identification of putative locations where the best matching region between a read and the reference is most likely to be found. To identify these locations, either the reference genome or the reads are split in small (12bp to 15bp) words and their positions are stored in a hash table. These hashed words are often referred to as seed words. Currently there are two types of seed words used. A seed word consist either of one consecutive genomic region (used in SSaha2 (Ning *et al.*, 2001)), or of a region where defined positions are not required to match named spaced seed words (used in SHRiMP2 (Rumble *et al.*, 2009)). After this initial search, a more accurate alignment, e.g. Smith-Waterman (Smith and Waterman, 1981) and Needleman Wunsch (Needleman and Wunsch, 1970), between the read and a smaller

subset of the before identified subregions is computed.

Both methods (BWT based and hash based) do not guarantee to identify the optimal alignment. However, typically both methods identify the optimal alignment region as the number of differences between the read and the optimal region on the genome is typically low.

2.2.2 De novo assembly

The main objective of de novo assemblers is an accurate reconstruction of the original sequence. However, the algorithms are mainly optimized to assemble the longest possible sequence stretch based on the reads. Figure 2.4 displays a general principles of the available methods. In contrast to previous methods that were designed for sequences of approximately 800bp in length (Pevzner *et al.*, 2001), current de novo assemblers are developed to process a high number of short sequences (100bp - 150bp). The majority of programs uses the modified De Bruijn graph (Pevzner *et al.*, 2001) where seed words build up a graph based data structure such that each seed word that overlaps with a different seed word given a read is connected (Figure 2.4 (a,2)). The nodes in this graph represent the seed words and are connected by edges, if they are observed overlapping each other. A set of connected nodes is often referred as a path

After such a graph is constructed an error correction is applied to reduce the number of alternative paths. Alternative paths are generated by similar regions that lead to two distinct sequences. This can either be because of sequencing errors or because of low complexity regions in the genome. To reduce the effect of sequencing errors, the frequency of all observed seed words is measured. If an alternative path exists of which the nodes have a reduced frequency, than the nodes of its counterpart path, the path is erased. For example, a path of the nodes (GCTGTAG) is detected that is supported by only 10 reads whereas a different path (GCTTTAG) exists that differs only by one mismatch and is supported by thousand of reads. In such a case, both paths are merged and the predicted sequencing error is corrected (Figure 2.4 (a,3 + 4)). In the end, consensus sequences are reported based on the corrected graph (Figure 2.4 (b)).

Advantages and disadvantages of de novo assembly were exhaustively discussed and described elsewhere. (Flicek and Birney, 2009; Zhang *et al.*, 2011; Narzisi and Mishra, 2011; MacLean *et al.*, 2009).

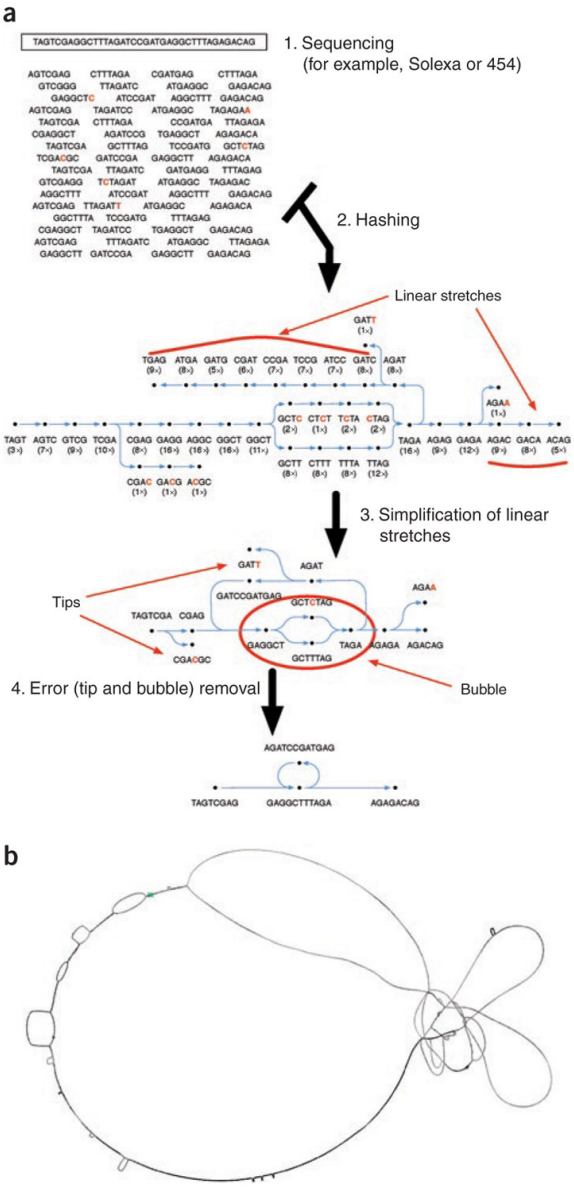

Figure 2.4: Schematic cartoon of a de novo assembly. (a) Simplified cartoon of the construction of a dependency seed word graph. (b) An example of one Bruijn graph of two related plasmids that share a common locus. The chosen seed word size was 30bp. The open loops reveal regions where the two plasmids are dissimilar. Image taken from Flicek and Birney (2009)

Chapter 3

NextGenMap: Next Generation Mapping

3.1 Introduction

The usage of Next Generation Sequencing (NGS) technologies allows for comprehensive analysis of genomes, transcriptomes, methylomes and epigenomes. Technologies such as Illumina or 454 have overtaken traditional sequencing strategies and generate data in the order of giga base pairs per machine and day (Metzker, 2010; Glenn, 2011). Enhanced resolution is accompanied by reduced costs per experiment compared to e.g. Sanger sequencing (Sanger *et al.*, 1977) methods. Therefore, NGS is becoming a widely used method in modern molecular biology. In all cases of an NGS experiment, one bioinformatic challenge concerns the mapping or assembling of the reads. Two strategies are in use: reference based mapping and de-novo assembly (MacLean *et al.*, 2009). The latter refers to sequence reconstruction without consulting already sequenced genomes. In contrast, reference based mapping relies on a closely related annotated genome sequence to which each read is aligned. An appropriate reference genome brings along the advantage of requiring a lower sequencing depth compared to de novo assembly. Moreover, this strategy is also used, when it comes to RNA sequencing to detect transcription factors or ChIP-Seq to detect binding sites where in general reads from a model organism are mapped to its genome. In summary, mapping approaches of reads provide a wide range of applications. However, if one wants to transfer genomic information from an available reference genome to a still unexplored organism, the performance of the available

methods is often not clear. Thus, it is necessary to understand the potential pitfalls and advantages of currently popular mapping programs. From a bioinformatics point of view, two essential criteria have to be taken into account when designing mapping algorithms for NGS data (Flicek and Birney, 2009). First, the sheer amount of data demands an efficient usage of computational resources. This includes optimization for runtime as well as memory requirements. Second, while lowering the computational burden of processing, the algorithm has to be able in most of the cases, to obtain the same results as an optimal alignment approach, i.e. handle mismatches, insertions, and deletions that may result from sequencing errors or evolutionary processes. The latter requirement becomes important when mapping reads from an unexplored and evolutionary distantly related species to well studied reference genome. Here, the currently popular mappers like Bowtie (Langmead *et al.*, 2009) or BWA (Li and Durbin, 2009) are not optimal as they are too conservative in their mapping process. To keep their computation time low (e.g. mapping a 10x human genome data set within hours), they typically allow just for a small number of mismatches. For a recent survey of available mapping application see Li and Homer (2010).

The ability to cope with a large number of mismatches, insertions and deletions during mapping becomes more pronounced when the read length is increased while the error percentage remains the same. This can be processed by using seed based mappers like SHRiMP2 (David *et al.*, 2011), which are typically slow. However, the here presented method called *NextGenMap* was designed to allow for a high number of differences while having a reduced runtime. We will show that *NextGenMap* outperforms currently available mapping programs with respect to mapping accuracy while maintaining the runtime at a competitive level.

NextGenMap utilizes high performance hardware such as a graphic processing units (GPU). The integration of GPUs as co-processing units allows for a computational speed up by exploiting widely available hardware. The parallelization of a GPU enables simultaneous calculations of several thousand computations per second (Vouzis and Sahinidis, 2011; Liu *et al.*, 2009; Trapnell and Salzberg, 2009). The substantial runtime reduction permits a more thorough sequence based search to identify the optimal alignment position, while still, outperforming ordinary CPU-based mapping algorithms. As the majority of desktop PCs are equipped with graphic cards, the mapping procedure can be deployed on standard workstations. Thus, the usage of *NextGenMap* as a standard

analysis tool offers a solution for institutes not equipped with expensive high performance computing infrastructure. To further facilitate the usage of desktop PCs, an user adjustable memory footprint is proposed that splits the data automatically into sub packages according to the available hardware. This enables the analysis of NGS data derived from, e.g. the human genome, on a standard PC. We introduce a novel evaluation method, that is used for the performance comparison of *NextGenMap* with currently available mapping programs. The evaluation allows to analyze the case when mapping data from organisms where the genomic sequence is unknown, to distantly related reference genomes. This method evaluates the accuracy of the mapping result based on a per base level for each read. In the following, we describe the *NextGenMap* approach. Then we will describe the results of a comparative simulation study, where we evaluate the performance of five state-of-art reference based mappers. Furthermore, we apply *NextGenMap* to real sequence data from *Arabidopsis thal.* and human.

3.2 Methods

3.2.1 NextGenMap

In the following we describe *NextGenMap*'s mapping strategy and discuss its performance. *NextGenMap* comprises three work-steps. First, it suggests candidate regions in the reference genome where a read potentially maps. This is done on the CPU. Second, for each candidate region *NextGenMap* computes a banded alignment score on the CPU or if available on the GPU. Third, for candidate regions with the highest alignment score per read, a banded alignment is computed on the CPU or GPU. A more detailed description and discussion about the computations of alignments on CPUs and GPUs can be found in chapter 4.

3.2.1.1 Candidate region search

The candidate search is based on small exact matching seed words of length k ($k \in \{2, \ldots, 32\}$). First, the reference genome is indexed, similar to other read mappers (Li and Durbin, 2009; Langmead *et al.*, 2009; David *et al.*, 2011), and a hash table is created. Each seed word is mapped to exactly one number, which is unique with respect to the

nucleotide sequence. Each genomic position of the seed word is stored in a k-hash table using a hashing approach. Next, for each read we extract all seed words. Given the list of seed words, all corresponding genomic position in the k-hash table are extracted. These positions are corrected by the location of the seed words in the read such that all point to the starting position of the alignment region. Due to insertions and deletions not all starting positions have to point at the same genomic region. Therefore, we introduce a spacing variable h, which defines the range in that all occurred starting positions are grouped together. Next we measure the number of observed seed words at a genomic position. If this number exceeds an user definable threshold w, we submit the genomic region and the corresponding read to the score computation.

3.2.1.2 SW alignment score computation

Here we compute the alignment score between a read and all its candidate regions, suggested in step 1. To improve runtime, and to achieve linear space requirement we compute the alignment scores only for a banded SW-alignment (Gusfield, 1997). The computational/storage complexity reduces to $O(l \times c)$ for the alignment matrix, where l is the read length and c is the corridor width. The parameter c specifies the maximal number of consecutive insertions or deletions. We note that the spacing parameter h, introduced in the previous section, is equal to $c/2$. Thus, the user specifies only the corridor width, c, and h is adjusted accordingly. Typically $c << l$. To find the region with the highest score, we do not need to store the full alignment matrix. By doing so, we can reduce the space complexity to $O(c)$ (Gusfield, 1997). The memory usage is now independent of the read length and depends only on the corridor width that reflects the length of the longest consecutive insertions or deletions. This reduction allows us to store the data on fast on-chip memory. Since we do not backtrack the optimal alignments, we reduce the branching behavior of *NextGenMap* that increases the efficiency of the GPU usage even more. In addition, the remaining variables are also stored in fast GPU memory. Only read and reference genome are stored in global memory. These reductions make it possible that each GPU thread computes one alignment score. Contrary to previous work (Vouzis and Sahinidis, 2011; Liu *et al.*, 2009; Trapnell and Salzberg, 2009), we can now compute millions of scores simultaneously on the GPU. Hence we use the full potential of the GPU.

3.2.1.3 SW alignment computation

For all region(s) in the reference genome that achieved the highest SW-alignment score, we finally compute the actual banded alignment. This computation including the back-tracking step take place on the CPU or on the GPU. Due to the efficient implementation, each thread computes one alignment. This step concludes the mapping of the reads. To reduce idle times of the CPU, while the SW scores and, subsequently the alignments, are computed on the GPU, the CPU computes in the mean time the candidate regions for the next reads. This guarantees maximal load of GPU and CPU.

3.2.2 Simulation and evaluation process

All mapping programs, including *NextGenMap*, are based on heuristic approaches that influences their mapping performance. To assess the performance of every mapping method, we measured the runtime and the mapping accuracy. We define mapping accuracy differently for real data sets and for simulated data sets. For real data sets, we simply give the number of mapped reads as we are lacking of a better metric. In case of the simulated data, we know where each read was generated from and its original alignment to the used reference sequence. Therefore, we can compute the number of correctly, wrongly and not mapped nucleotides.

Figure 3.1 summarises the simulation and evaluation pipeline. Our simulation process consist of two stages. First, a reference chromosome is read in. This is altered by random mutations. An user defined parameter gives the probability of observing a mutation at each position. If a position is altered it is decided if it is a mismatch or an insertion and deletion (indel). The probability of an indel is 10% given a mutation, for simplicity the indel length is fixed by 1. The probability for an insertion or deletion are both 50% given the event of an indel. For each mutation event the probability of all alternative nucleotides compared to the original in the reference are equal. This mechanism follows a Juckes Cantor (Jukes and Cantor, 1969) model except we also simulate insertions and deletions. If a mutation takes place, we track the type of mutation plus its location on the reference genome. Second, we simulate the process of sequencing by a simplified model. We randomly select locations across the altered genome as starting positions of reads. For each sampled read, we run over the sequence. Given an user defined probability, every position has the chance to become a sequencing error. If the event of a

sequencing error occur, we follow the same strategy as described above given a mutation. In addition, we store every sequencing error that altered the read sequence. In the end, we store in one file the resulting read sequence and additionally the true alignment given the original reference sequence. The latter is used to evaluate the mapping of the reads. Note that the sequencing error will not affect any other read, whereas a mutation will be present in every read that overlaps with the mutated position.

In the evaluation procedure we then mapped the simulated reads to the original reference sequence and compare the results to the previously stored true alignments. For each read we compute the fraction of correctly placed (p_C), wrongly placed (p_F), and not placed (p_{NP}) nucleotides, with respect to the reference genome. The mapping of each read can therefore be characterised by these three fractions, which naturally sum up to one. The fractions (p_C, p_F, p_{NP}) can be viewed as a point inside an equilateral triangle (Cannings and Edwards, 1968; Strimmer and von Haeseler, 1997). For a given point inside the triangle, the p's correspond to the length of the perpendiculars from the point to the sides of the triangle. If one of the p's is close to one, then the point will be close to a corner of the triangle. Thus, each corner represents an extreme case, where a read is not mapped at all (left corner), entirely wrongly mapped (right corner) or correctly mapped (upper corner). To illustrate this, assume that the alignment of a read consists only of correctly mapped nucleotides and wrongly mapped nucleotides then the resulting point (p_C, p_F, 0) would fall onto the right side of the triangle. The centre of the triangle describes the situation where one third of the read is correctly, one third is wrongly and one third is not mapped at all. This illustration provides a comprehensive view of the performance of the reference mapping programs. To account for the different coverage of points inside the triangle, we introduce a color density gradient that reflects the percentage of reads falling in each area.

3.2.3 Comparative study of reference based mapping methods

We selected five of the most frequently used programs to evaluate their performance and to compare them to *NextGenMap*. Bowtie (Langmead *et al.*, 2009), BWA (Li and Durbin, 2009) and BWA-SW (Li and Durbin, 2010) are chosen as representatives for the Burrows Wheeler based approaches. In addition, SSaha2 (Ning *et al.*, 2001) and SHRiMP2 (David *et al.*, 2011) are selected as programs based on local sequence

alignment. Every mapping program was tuned to exploit the available hardware as good as possible. The measured runtime (wall clock time) includes indexing and mapping time for all programs as these steps define a full mapping process. For *NextGenMap* we measured the wall clock time with (*NextGenMap*+GPU) and without the graphic card (*NextGenMap*). All programs were executed with the default, respectively with the recommended parameter settings (see chapter 3.2.5). All programs were executed on the same PC, equipped with an AMD Phenom II X4 965 Quard-Core with 16 GB of memory and a GTX 480 graphic card.

3.2.4 Real data

The human data set consisted of 10 million reads (100bp long) randomly sampled from SRR064182 (NCBI Sequence Read Archive). We used the human genome (GRCh37) as a reference genome to map the reads with the individual mapping methods. The *Arabidopsis thaliana* data consists of 14 million sequencing reads (36bp long) from strain Bur-0 (Ossowski *et al.*, 2008). The reference sequence of strain Col-0 was used for mapping the reads.

3.2.5 Parameter settings

NextGenMap comes with default settings for short (< 150bp) and long (> 150bp) sequences. In addition, *NextGenMap* includes preconfigured parameters that specify the sensitivity of the search (for example, the number of mismatches to expect). This parameter has three levels (low, normal, high). For the real data sets and the majority of the simulated data sets (S_1, S_2 and S_3) this parameter were set to normal. Whereas in the case of S_4, we set the parameter to enable a more throughout search to high.

For data sets with short sequences *NextGenMap* uses by default a seed word size of 12 and require minimum two exact matches of seed words between a read and a region of the reference to trigger the SW alignment score calculation. Seed words were extracted from the reference at each position and from the reads at every second position, except for dataset S_4 where *NextGenMap* used an increased sensitivity parameter that extracted seed words starting at every position from the reads. The Smith-Waterman scoring parameters were set to $+5$ for a matching base pair, -2 for a mismatch, and -5

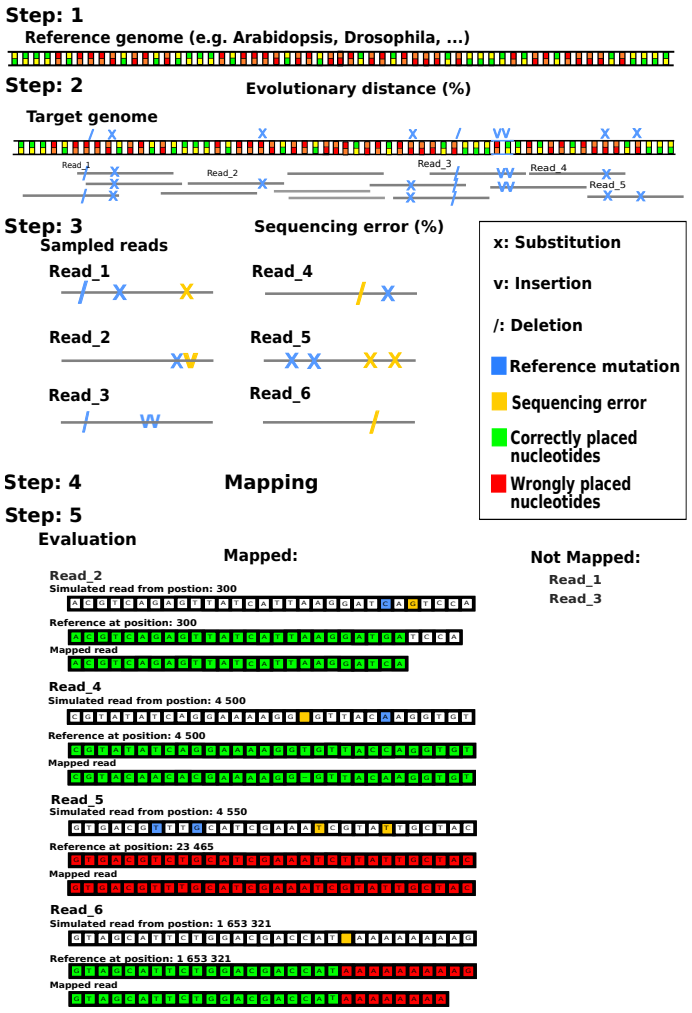

Figure 3.1: Simulation schema: Step 1: A reference genome is read in. Step 2: a target genome sequence is simulated from the reference genome by randomly positioning mismatches, insertions and deletions. Step 3: sequence reads are simulated by randomly sampling sequences of a fixed length from the target genome. Each read is then subject to a possible sequencing errors. Thus, each read might differ from the corresponding region in the reference genome either due to the evolutionary process (blue) or to sequencing errors (orange). Step 4: The reads are mapped to the reference genome using the original reference sequence. Step 5: The reads are evaluated based on the true alignments.

for insertions or deletion. The width of the banded Smith-Waterman alignment was set to 20; allowing for a maximum of 10 consecutive insertions or deletions.

The other programs were executed with the respective recommended parameter settings. However, some adaptations were necessary: For data set S_4 we ran Bowtie with "-l 20 -n 3 -e 400 -a –best -strata", which increased the performance in terms of mapped reads. For the other data sets Bowtie was executed with "-S -f" parameters. The parameters of BWA were adjusted using: "-k 3 -l 20 -n 9 " for S_4 to achieve more mapped reads. SSaha2 was executed with " -solexa -best 1 -cut 2000 ". SHRiMP2 was executed with " -strata -local ".

3.3 Results

Program	human			*Arabidopsis thal.*		
	M (%)	U(%)	runtime (min)	M (%)	U (%)	runtime (min)
Bowtie	50.1	50.1	164 min	66.4	66.4	**7**
BWA	69.4	66.8	153 min	66.9	54.3	9
BWA-SW	91.6	88.6	**124 min**	54.9	40.1	14
SSaha2	-	-	$>$ 4 days	93.8	68.1	512
SHRiMP2	93.4	62.4	65 hours	85.7	64.4	34
NextGenMap	**94.2**	**88.9**	587 min	**97.3**	**75.8**	19
NextGenMap $_{(+GPU)}$	**94.2**	**88.9**	417 min	**97.3**	**75.8**	16

Table 3.1: Percentages of mapped (M) and uniquely mapped reads (U) for human data (10 million reads; read length of 100bp) and *Arabidopsis thal.* Bur-0 data (14 million reads; read length of 36bp) mapped to *Arabidopsis thal.* Col-0. The data sets differ since the mapping of *Arabidopsis thal.* includes a $\sim$ 5 times higher evolutionary distance between the sequenced and the reference sequence compared to the human data.

3.3.1 Evaluation of a real data set from human

Here we evaluate the performance of *NextGenMap* on real data when reads and reference genome are both from human. Table 3.1 shows the percentages of mapped and uniquely

mapped reads for different mapping programs, when aligning 10 million human reads. BWA-SW was the fastest mapper (134 min), followed by BWA and Bowtie requiring 153 and 164 minutes, respectively. SHRiMP2 had the longest runtime (65 hours) and mapped 93% of the reads. *NextGenMap* mapped 94% of the reads within 417 minutes using the graphic card and under 10 hours without using the graphic card. This data set represents a standard case of mapping reads to a reference. Next we will have a look at the performance of the different mappers if we increase the evolutionary distance between the reference genome and the sequenced reads.

3.3.2 Evaluation of a real data set from *Arabidopsis thaliana*

To further evaluate the performance of *NextGenMap* and other mappers on data, that now show a lower extend of similarity between the read and the reference, we used the *Arabidopsis thaliana* strain Col-0 as reference genome and reads from *Arabidopsis thal.* strain Bur-0 as target genome (Ossowski *et al.*, 2008). The evolutionary distance between both strains is about 0.6% (Nordborg *et al.*, 2005). This is about 5 times the evolutionary distance between two human individuals. We aligned 14 million Bur-0 reads of length 36 (Nordborg *et al.*, 2005) to the reference genome Col-0 (genome size about 157 million base pairs). Table 3.1 shows the percentage of mapped and uniquely mapped reads. *NexGenMap* mapped the most reads (97.3%) followed by SSaha2 (93.8%), SHRiMP2 (85.7%) and BWA-SW (54.9%). BWA and Bowtie mapped only two thirds of the reads (67% and 66% respectively). The Burrows Wheeler based approaches require to set an upper limit on the number of mismatches, insertions and deletions to the reference genome per read. If this bound is exceeded, then they cannot map the read, an event that is likely given the increased evolutionary distance (0.6%) to the reference. Since BWA-SW is designed for long reads ($> 100bp$) it can not cope with this data set. In terms of runtime, *NextGenMap* (16 min and 19 min with CPU only) beats SSaha2 (521 min) as well as SHRiMP2 (34 min). Bowtie and BWA run significantly faster (7 min, 9 min), but only mapped two thirds of the reads. BWA-SW required 14 minutes. Thus, an evolutionary distance of 0.6% between target genome (Bur-0) and reference genome (Col-0) already leads to a dramatic drop of the number of mapped reads for Bowtie and BWA. The run-time advantage of Bowtie and BWA comes with the expense of loosing many reads, compared to the other SW-based mappers. We also note that the fraction of (uniquely) mapped reads provides only a very simplified picture of the performance

of NGS mappers. Important information like the number of reads that are wrongly mapped or the number of reads that are only partially mapped is missing. However, this information would help to provide a more detailed description of the mappers. To further elucidate the effect of the evolutionary distance of a target genome to its reference genome, we carried out a detailed simulation study.

3.3.3 Simulated data sets

We simulated three Illumina type data S_1, S_2, and S_3 with read lengths of 36 bp, 72bp, and 150 bp, respectively. We assume that the target genome has an evolutionary distance of 1% to the reference genome. This represents for example the situation when using a closely related species, such as chimp and human as target and reference species, respectively. Alternatively it resembles the case where the evolutionary distance within a species is around 1%, e.g. as it is the case in *Candida albicans* (MacCallum *et al.*, 2009). A fourth simulation (S_4) covers a more extreme scenario, where the targeted genome has an evolutionary distance of 10% to the reference genome. Table 3.2 summarizes the simulation settings. For *Arabidopsis thaliana* we simulated a 15 fold and for *Drosophila melanogaster* a 20 fold coverage.

ID	Reference organism (chr 1)	Evolutionary distance(%)	Read length	Sequencing error(%)	Number of reads
S_1	*Arabidopsis thal.*	1	36	2	12.6
S_2	*Arabidopsis thal.*	1	72	2	6.3
S_3	*Drosophila mel.*	1	150	2	3.1
S_4	*Drosophila mel.*	10	72	2	6.4

Table 3.2: Overview of the simulated data sets.

Table 3.3 summarizes the percentages of mapped and uniquely mapped reads for the four simulation scenarios and for different mapping programs. While the alignment based methods (SSaha2, SHRiMP2, NextGenMap) show a high percentage of mapped and uniquely mapped reads regardless of read length and evolutionary distance to the reference genome, the Burrows Wheeler based mapping programs are greatly affected by read length and evolutionary distance as shown by the decline of the fraction of mapped reads. A more detailed analysis of the mapping results are shown in Figure

3.2. The percentages at the corners indicate the fraction of reads that represent three extreme cases, i.e. reads that are entirely correctly mapped (top corner), not mapped (left corner) or mapped entirely wrong (right corner).

Program	S_1 (%)		S_2 (%)		S_3 (%)		S_4 (%)	
	M	U	M	U	M	U	M	U
Bowtie	80.8	80.8	52.4	52.4	15.1	15.1	33.1	33.1
BWA	90.1	84.8	87.4	83.3	77.6	75.7	55.5	53.8
BWA-SW	56.1	52.2	93.3	89.3	**100.0**	**95.5**	35.4	33.4
SSaha2	98.6	91.4	**100.0**	**95.5**	**100.0**	95.2	96.8	88.2
SHRiMP2	99.3	92.8	**100.0**	**95.5**	**100.0**	95.2	93.3	88.3
NextGenMap	**99.8**	**93.2**	**100.0**	**95.5**	**100.0**	95.2	**100.0**	**94.4**
NextGenMap $_{(+GPU)}$	**99.8**	**93.2**	**100.0**	**95.5**	**100.0**	95.2	**100.0**	**94.4**

Table 3.3: Percentages of mapped reads (M) and uniquely mapped reads (U) for simulated data given the outcome of different mapping programs.

3.3.3.1 S_1: 36p reads from a close relative of a model organism

Column 1 in Figure 3.2 displays the performance of the mapping programs for S_1. Overall the mapping accuracy is high and all programs map only a small fraction of the reads to the wrong position on the reference genome (2-4%). The main difference lies in the fraction of unmapped reads. BWA-SW and the Burrows Wheeler based methods (Bowtie, BWA) display a relatively high fraction of not mapped reads (44%, 19% and 10%, respectively), while the local alignment based methods show only 1% of not mapped reads. Except for BWA-SW, all methods map at least 70% of all reads entirely correct. Next, we analyzed the accuracy of the mapping. All methods place most of the reads with a high fraction of correctly mapped nucleotides, as indicated by the deep red color at the top corner of the triangles. Among these, however, *NextGenMap* has with 83% of entirely correctly placed reads the highest accuracy. The computing times range from three to six minutes for the Burrows Wheeler based approaches. *NextGenMap*+GPU takes 7 minutes (8 minutes when using the CPU), BWA-SW takes 11 minutes and SHRiMP2 takes 12 minutes performing the mapping. SSaha2 is with 117 minutes by far the slowest program in this study. In summary, for simulation data set S_1. *NextGenMap* shows the best performance with respect to the number of correctly mapped reads.

3.3.3.2 S_2: 72bp reads from a close relative of a model organism

Simulation S_2 (Figure 3.2, Column 2) differs from S_1 by using twice as long reads. Note that the average percentage of differences (mutations and sequencing errors) is 3% and therefore the same as in S_1. On the one hand, greater read length brings along a higher number of differences between the read and the reference genome. On the other hand, it reduces the chance that the read is wrongly mapped. The simulation confirms our intuition; the percentage of wrongly mapped reads drops (2%) for all methods when compared to S_1. Due to the increased number of differences between read and reference genome, the Burrows Wheeler based approaches show an increase in the percentage of not mapped reads. For Bowtie 48%, and for BWA 13% of the reads were not mapped. BWA-SW fails to align 7% of the reads. The other programs map all the reads to the reference genome. Except for BWA-SW, the number of entirely correctly mapped reads is reduced for all methods. However, still SSaha2 (72%), SHRiMP2 (75%), and *NextGenMap* (81%) map more reads correctly than BWA-SW (63%). Runtime wise, the Burrows Wheeler based approaches (Bowtie, BWA) performs best, however, again they have the shortest runtimes at the expense of a large fraction of not mapped reads. *NextGenMap* shows the best performance with respect to the number of entirely correctly mapped reads while having a competitive runtime (4 minutes on the GPU and 6 minutes for the CPU version). BWA-SW and SHRiMP2 required 16 and 11 minutes respectively, to map the reads. Again SSaha2 has the longest runtime (47 minutes).

3.3.3.3 S_3: 150bp reads from a close relative of a model organism

In simulation S_3 (Column 3, Figure 3.2) the read length is again doubled and equals to 150bp. In this simulation, we observe the same trend as in S_2 but now on a different reference sequence; Bowtie fail to map a large portion of reads (85%), while BWA showed only a reduced capability to map the reads (22% not mapped). BWA-SW and the other local alignment based methods map all the reads. In terms of entirely correctly mapped reads, *NextGenMap* showed again the best performance with 79% reads followed by SHRiMP2 with 74% of correctly mapped reads. As before, Bowtie (2 minutes) and BWA (5 minutes) delivered again the fastest runtime. However, *NextGenMap* succeeded in mapping 79% of the reads entirely correctly within 2 minutes (4 minutes without using the graphic card). The remaining mapping programs required 11 to 18 minutes to finish

the mapping.

3.3.3.4 S_4: 72bp reads from a distant relative of a model organism

The target genome in simulation S_4 has an evolutionary distance of 10% to the reference genome. Using default settings of Bowtie and BWA they both map fewer than 5% of the reads. To allow Bowtie and BWA to stay competitive, an adequate adjustment of the parameters was done. As a result Bowtie and BWA map 33% and 55% respectively. However, this adjustment increased the computing time considerably. Figure 3.2, column 4 shows the results. The mapping with Bowtie needed 18 minutes, while BWA lasted about 8 hours. With the exception of BWA-SW, the alignment based programs map a large fraction of reads. This shows the advantage of SW alignment based mapping approaches compared to Burrows Wheeler based approaches. However, the number of entirely correctly mapped reads drops dramatically. *NextGenMap* shows a much smaller variation compared to the other SW based methods, the yellow and red colors are concentrated at the top corner. The color gradient shows that SSaha2 and SHRiMP2 deliver a high fraction of partially correctly mapped reads with a relatively large variation. *NextGenMap* shows with 46% of entirely correctly mapped reads the best performance. However, the color distribution indicates that *NextGenMap* has a tendency to map nucleotides to the wrong positions rather than not mapping them (colored area tends to the right). SSaha2 and SHRiMP2 are more conservative by not mapping the nucleotides (colors tend to the left). In summary, *NextGenMap* maps more reads than all other programs considered while maintaining a low number of wrongly positioned reads. This even holds for a large evolutionary distance between reads and reference genome.

3.4 Discussion

The first crucial step in the reference sequence based analysis of Next Generation Sequencing data is the mapping procedure. All subsequent analyses are influenced by the outcome of the mapping. Different sources of sequence difference between sample and reference sequence play a role. First of all, any kind of sequenced genome exhibits a certain amount of variation compared to the available reference sequence. Secondly,

the reads themselves may contain errors introduced by e.g. sequencing or base calling errors. Thus, there is a high probability that a sequence read is not 100% identical to the corresponding genomic region in the reference and the probability increases with increasing genetic distance between target and reference genome, read length and sequence quality. Still, the most frequently used assembly/mapping programs handle deviations between read and the reference genome by just allowing a maximal number of deviations (nucleotide differences) and, if supported, gaps. It is, thus, an improvement of *NextGenMap* to be equally good applicable over all analyzed read lengths, making our program the most versatile mapper. That is *NextGenMap* does not restrict the alignment via a threshold on the number of mismatches or gaps. *NextGenMap* enables an user-defined mapping stringency via defining the seed word length, the alignment scoring function and the adjustable banded alignment. By allowing a higher flexibility, we map more reads correctly to their corresponding genomic position compared to standard mapping algorithms. This becomes especially relevant when the target and reference genome are rather distantly related. As a consequence of the more comprehensive mapping it is possible to reduce the sequencing depth, or the increased number of mapped reads will result in more statistical power to detect SNPs for instance. Our evaluation on real data demonstrated the capability of NextGenMap. Here, *NextGenMap* beats the second best mapping program (SHRiMP2) by mapping 1% more reads for the human data and by 3% more reads for the *Arabidopsis thal.* data, respectively. The reduced capability of BWA-SW when aligning short reads is known (Li and Durbin, 2010) and therefore not taken into account. Bowtie performs good on short reads while BWA-SW shows a reduced capability to align short reads. In contrast, BWA-SW becomes more reliable for longer reads while Bowtie has problems to align longer reads. *NextGenMap* shows equally good applicability over all analyzed read lengths, making our program the most

Comparing to BWA or Bowtie, *NextGenMap* aligned 1/3 more reads. The most striking strength of *NextGenMap* is demonstrated by the simulated data set with a higher evolutionary distance (S_4), where *NextGenMap* mapped 49% of all reads, whereas other programs suffer severely from such an evolutionary distance. Thus, the real strength of *NextGenMap* becomes apparent, when one expects large differences between the reads and the reference genome. This is the case, if one wants to sequence a genome from a taxon that is distantly related to a neighbour taxon, of which the genome sequence is already available.

Apart from the increased sensitivity a major strength of *NextGenMap* lies in exploiting the available resources of a PC. Using a graphics card as co processor allows an extensive search for putative alignments without dramatic increase in runtime. Porting Smith-Waterman local alignment on the GPU has been reported before (Vouzis and Sahinidis, 2011; Liu *et al.*, 2009). However, the authors focused on a parallelization of one SW alignment computation rather than calculating several hundreds SW alignments in parallel as done here.

While technical improvements will certainly help to speed up mapping attempts, it is also necessary that the advancement in speed is used to increase the applicability of mapping approaches including the de novo sequencing of non model organisms. *NextGenMap* fills this niche. Even if it may not be possible to reconstruct the large scale organization of an unknown genome, the knowledge of the annotated genome of a model organism together with *NextGenMap* may provide first insights into the gene content of a non-model organism. *NextGenMap* will, depending on the sequencing depth also provide local information about genomic architectures. Therefore, mapping approaches may serve as a better strategy to gain insights into genomes than de novo approaches. This provides a quick and efficient knowledge transfer on the gene level of model organisms towards non-model organisms without studying or knowing their genomes.

To evaluate the different mapping programs, we introduced a novel evaluation schema together with a visualisation of the performance of the programs. We are aware that the simulation scheme is quite simple and could be easily extended using tools from molecular evolution, e.g. Dawg (Cartwright, 2005). For the time being, we believe that a simple Jukes-Cantor model (Jukes and Cantor, 1969) is sufficient to mimic important sources of genomic variability, if organisms are not too distantly related. Although we did not apply an advanced sequencing error model, we believe that we could mimic the general picture of the error sources. More in depth modelling of the individual error sources is possible but was not the scope of this work. The simulations already show the advantages of our approach with respect to speed and of the SW-alignment approach with respect to mapping accuracy. Moreover, SW brings along the advantage of being a well-understood method that offers the possibility to discuss the alignment significance in a statistical framework. This allows to evaluate alignments not only based on heuristics but on proper statistics.

3.5 Conclusion

NextGenMap can efficiently process deep sequencing data. Furthermore, we demonstrate that *NextGenMap* maps successfully reads with relatively large evolutionary distance between the targeted genome and the annotated reference sequence. This and its usage of graphic cards make *NextGenMap* a flexible tool to transfer knowledge from model to non-model organisms.

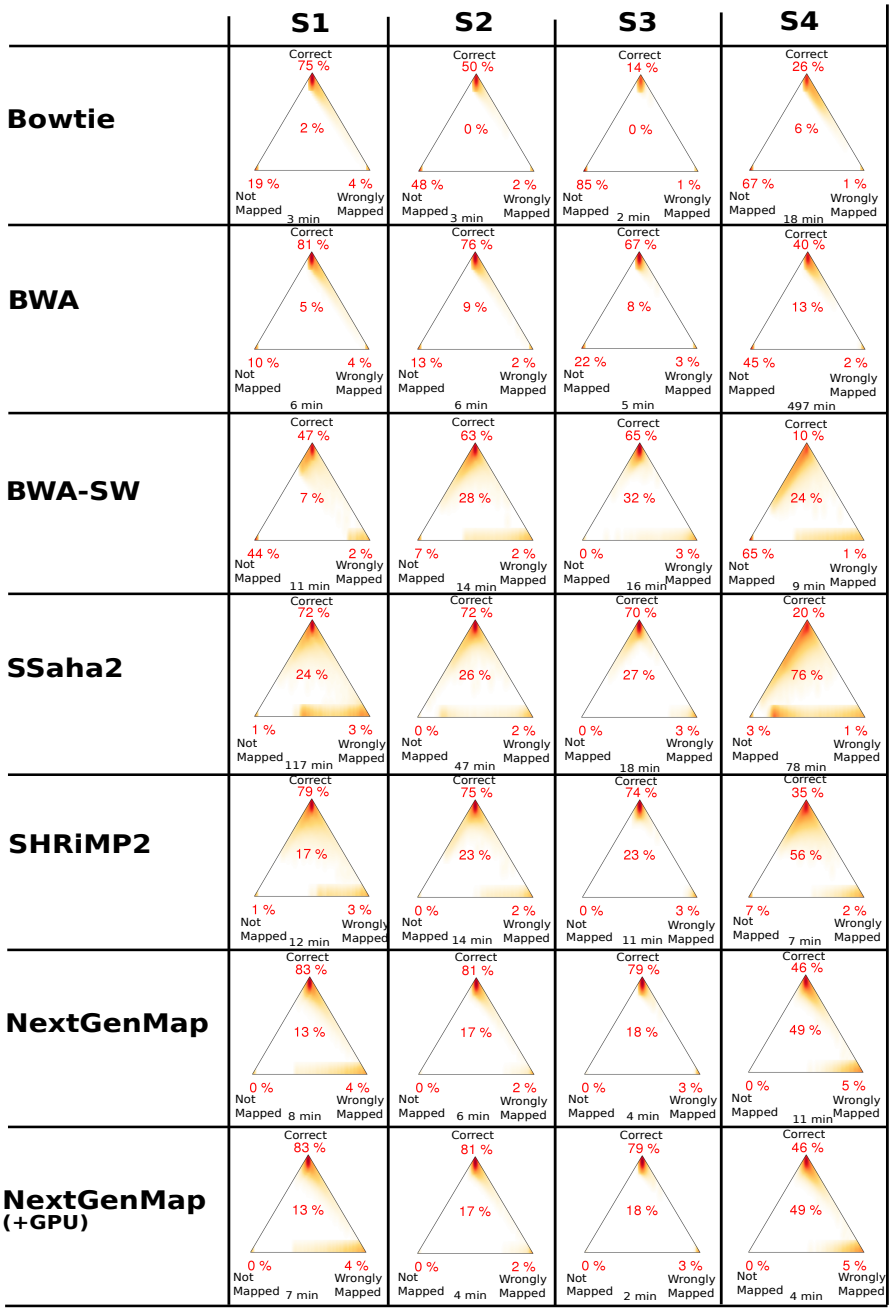

Figure 3.2: Results of selected mapping programs for simulation settings S_1 to S_4 as described in Table 3.2. The percentage in the center summarizes the fraction of the reads, where the relative frequencies of correct, wrong and not mapped nucleotides are always less than one. The other numbers represent the percentages of the reads that fall onto the corners of the triangle. The evaluation was carried out only on uniquely mapped reads. The runtime (in minutes) of each program is depicted below the triangle.

Chapter 4

MASon: Million Alignments within Seconds

4.1 Introduction

Analyzing NGS data is a demanding task. Keeping pace with the ongoing technological developments poses a great challenge to bioinformatics. The first step when analyzing NGS data is mapping reads to a known reference sequence. In the previous chapter, we introduced a way to map reads using a seed word based approach in combination with a local alignment approach. To guarantee a fast and reliable search for (putative) mapping positions, we optimized the pairwise alignment method for GPUs and CPUs. In this chapter, we will go into the details of the optimizations and of the implementations of a Smith-Waterman based alignment algorithm on various high performance hardware platforms.

The general problem that arises when mapping millions of reads to a large reference genome is that the number of required alignment computations increases. Typically the computation of the alignments consumes the main part of the runtime of the mapping software. To reduce the number of required alignments, every mapping method including NextGenMap applies a filtering step based on the information of the candidate search (see chapter 3 for details). When a more stringent filtering step is applied, the number of required alignments per read is close to one. This results in a fast runtime and typically a high percentage of reads are still mapped. However, if one is interested in accurately aligning reads also to high polymorphic regions, one requires a relaxed filtering strategy,

and therefore the number of alignment computations again increases. This typically results in an increased runtime of the mapping methods. The objective of this chapter is to provide a solution to this problem that accomplish a high number of alignments while achieving a low (within second(s)) runtime.

As mentioned before, the general problem is to compute millions of alignments. There are two possibilities to reduce the overall runtime. One possibility is to utilize available hardware by parallelizing the alignment problem or by using SIMD (single instruction multiple data) instructions. Previous approaches, e.g. CUDASW++ (Liu *et al.*, 2009), GPU-Blast (Vouzis and Sahinidis, 2011) or striped Smith-Waterman (Farrar, 2007), focused on improving the performance of one single alignment calculation. Contrary to these methods we focus here on a high number of local alignments to be simultaneously calculated.

The other possibility is to adapt the alignment algorithm to the underlying problem. Several optimization strategies are known for pairwise alignments. In this chapter, we will combine some of these to optimize a Smith-Waterman local alignment algorithm for NGS data. Since sequencing technologies show a decline in accuracy at the end of the read (Harismendy *et al.*, 2009), a local pairwise alignment (Smith and Waterman, 1981) approach is especially suited for this, since it terminates the alignment as soon as too many mismatches, insertions or deletions occur.

In the following we will describe and discuss the optimization strategies. We will describe strategies for porting the optimized alignment implementations on Nvidia's Compute Unified Device Architecture (CUDA) (Kirk and Mei, 2010), the Open Computing Language (OpenCL) (Stone *et al.*, 2010) and Streaming SIMD Extensions (SSE) (Raman *et al.*, 2000). The results will be compared to a CPU implementation of the optimized algorithm and to an optimal Smith-Waterman algorithm. Finally, we measure the performance on different platforms and discuss their advantages and disadvantages.

All optimizations and implementations are provided in a C++ library called MASon. We focus on the interfaces for an easy integration and usage of MASon in existing mapping methods. In addition, MASon is designed to automatically adjust itself to the underlying hardware in terms of memory and number of available computing cores.

4.2 Methods

4.2.1 Programming interfaces

To fully utilize every hardware platform ranging from desktop CPUs and GPUs to highly parallel server processors, we are using different application programming interfaces (APIs).

4.2.1.1 SSE

The Streaming SIMD Extension (SSE) (Raman *et al.*, 2000) is a single instruction multiple data (SIMD) instruction set. SSE was introduced by Intel with the Pentium III processor. Today, SSE is available on virtually all computers. The SSE specification consists of several vector instructions that execute a single operation on four different floating point variables in parallel. Originally, SSE was designed to speed up single precision floating point operations as they occur in image or signal processing. Newer versions of SSE also support instructions on other data types.

4.2.1.2 CUDA

The Compute Unified Device Architecture (CUDA) (Kirk and Mei, 2010) is a C/C++ parallel computing architecture developed by Nvidia. It enables developers to write scalable C code in a CPU like manner that can be executed on a graphic processing unit (GPU). In contrast to CPUs, GPUs have an architecture that is designed to maximize instruction throughput. This is achieved by a high number of streaming processing units that are arranged in multiprocessors. Every graphic card has global memory that is comparable to the RAM for the CPU. When running computations on the GPU all data has to reside in this global memory, which typically has 1-2 GB available. To gain performance, they can later be split and moved to more efficient memory types of the GPU. These memory types are located directly on each multiprocessor and are orders of magnitudes faster than global memory. However, they are typically very small (16-48 kb per multiprocessor).

4.2.1.3 OpenCL

The Open Computing Language (OpenCL) (Stone *et al.*, 2010) is an open standard that provides an unified programming interface for heterogeneous platforms including GPUs and multi core CPUs. In contrast to CUDA, OpenCL is not limited to a single hardware manufacturer. OpenCL provides an API that enables software developers to take advantage of the parallel computation capabilities of modern hardware, such that a single implementation runs on a wide range of different platforms. The execution as well as the memory model is very similar to CUDA.

4.2.2 Implementation

Computing Smith-Waterman (SW) alignments is very demanding in terms of runtime and memory resources. The time as well as the space complexity is $O(|| R || \times || G ||)$. For NGS, R is the read length and G is typically the length of a small sub region of the genome identified by an exact matching approach (Ning *et al.*, 2001; Rumble *et al.*, 2009; Homer *et al.*, 2009). $|R|$ is typically small, ranging from 36 up to 150 bases for Illumina sequencing. However, usually there are millions of reads produced by each experiment. For example, when sequencing the human genome with 20-fold coverage using Illumina technologies (150bp) the number of reads m is ca. 400 million. In addition, exact matching approaches usually report more than one possible mapping regions per read. For example, when mapping to the human genome, using an exact matching substring of 12, the number n of putative subregions per read is $n = 180$. Thus, the number of alignments ($m \times n$) that has to be computed is 72 billion. To deal with this large number of alignments, one needs to employ available hardware as efficiently as possible. For NGS data parallelizing the computation of a single alignment is not useful because the computational cost for a single alignment is low. Therefore, we use a single thread to compute one alignment and focus on computing as many alignments in parallel as possible. This brings along the problem that every hardware platform has different capacities when it comes to parallel computation. On a singe CPU one can only use a small (4-8) number of threads. In contrast, a GPU is only working efficiently when running a high number (thousands) of threads in parallel. This means that a minimum number of alignment computations (batch size) is required to fully utilize a GPU. However, this minimal number varies between GPUs. To ensure

that our implementation fully loads different GPUs, independent of their architecture, we calculate the batch size b:

$$b = N_{mp} \times t_{max} \tag{4.1}$$

where N_{mp} is the number of multiprocessors, t_{max} is the maximum number of threads per multiprocessor. We furthermore require that

$$b \times M_{align} \leq M_{avail} \tag{4.2}$$

where M_{align} is the memory required per alignment and M_{avail} is the available global memory on the graphic card (Nvidia, 2009) and b is chosen accordingly to equation 4.2. If $b \times M_{algin}$ exceeds M_{avail}, b is reduced accordingly. For modern GPUs the batch size ranges from 50,000 to 100,000 alignment computations. This high numbers cause the problem that the memory available for computing a single alignment is small. Thus, we implemented a SW alignment that is less memory demanding. To achieve an efficient solution, we make one prior assumption. Most NGS projects sequence individuals from a species for which a reference genome already exist, e.g. 1000 Genomes Project Consortium (2010), or they sequence a new species that is closely related to a species for which a reference genome already exists. Thus, we assume that the number of insertions and deletions between the read and a genomic regions are typically small. We implemented a $k-$difference alignment, also called banded alignment algorithm citepGusfield1997. Here, only a small corridor surrounding the diagonal from upper left to lower right of the matrix H is computed. Thus, the complexity is reduced to $O(|| R || \times c)$, where c is the corridor width and $c << || G ||$. The memory reduction allows the computation of a high number of alignments in parallel such that the GPU is fully loaded. However, due to its size even the reduced matrix H has to be stored in global memory. Since the global memory is slow the computing time would be still too high to compute alignments for all putative genomic regions. To further reduce the computing time, we take into account the observation that in NGS mapping only the alignments between a read and the sub regions of the genome that show the highest alignment scores are of interest. To identify the best matching sub region in the first place, the optimal alignment score is sufficient. Thus, the matrix H can be reduced to a vector of length c and the space complexity is reduced to $O(c)$ (Gusfield, 1997). Note that the memory usage of score computation is independent of read length. However, the computational complexity remains $O(|| R || \times c)$ and the corridor width c will indirectly depend on the read length,

since it determines the number of consecutive insertions and deletions. The reduced memory requirements allow us to store H in fast on-chip (shared) memory. In addition, all remaining variables, like the scoring function, are also stored in fast memory (private, constant and texture). Only read and reference sequences are stored in global memory. This minimizes the access to global memory. Our approach works for all corridor sizes smaller than 100 on current GPUs (2011). Since the length of indels is typically small this restriction of the corridor size is not likely to limit the applications.

Although GPUs are powerful co-processing units, they are not available on every computer. In such cases a fast computation of alignments on CPUs is desirable. Except for multi-threaded programming, the most common mechanisms for parallelization on the CPU is SSE. As mentioned before, SSE allows the programmer to execute a single instruction on multiple values. The data type that is used determines the number of concurrently processed instructions. Since we are working with sequences shorter than 2000 base pairs, 16 bits are sufficient to store the alignment score. Thus, we can store eight values in the 128 bit wide XMM registers. This speeds up score calculations by processing eight alignments in parallel. However, since SSE only supports a limited set of instructions (in our case: compare, add and max), the observed speedup will be lower than eight.

OpenCL offers another possibility to increase performance on the CPU. As mentioned, OpenCL device code is independent of the hardware. However, to achieve maximal performance it is necessary to optimize the implementation of the SW algorithm for the different platforms (CPU and GPU). The most important difference between both platforms is that on the GPU it is necessary to use optimized memory access patterns (Nvidia, 2009), whereas on the CPU these patterns would hinder an optimal caching and therefore degrades performance. The second major difference is that on the CPU the OpenCL compiler implicitly uses SSE instructions to increase performance. However, this is only possible when using specific (vector) data types, which cannot be used efficiently on a GPU. Besides these two points the OpenCL code we use for GPUs is identically to the one we use for CPUs.

To allow NGS application programmers to employ these technologies, we created a software library that consists of three implementations. (1) A CPU version including SSE instructions, (2) a CUDA based version for Nvidia graphic cards and (3) an OpenCL implementation suitable for GPUs and CPUs. All of them are encapsulated in dynamic

		Score computation (Alignment computation)							
		$c = 15$				$c = 30$			
Implementations		36	72	100	150	200	500	700	1000
CPU	classical SW	- (9.67)	- (33.56)	- (61.42)	- (132.12)	- (229.19)	- (1420)	- (2737)	- (5602)
CPU	SeqAn	- (6.77)	- (12.82)	- (17.49)	- (25.87)	- (64.08)	- (159.59)	- (223.53)	- (320.04)
CPU	banded SW	3.00 (6.73)	5.95 (12.72)	8.26 (17.33)	12.68 (25.64)	34.61 (63.75)	81.94 (153.88)	114.73 (214.79)	164.09 (305.92)
CPU	+ SSE	**0.98** (6.73)	**1.95** (12.72)	2.71 (17.33)	4.06 (25.64)	**10.23** (63.75)	26.37 (153.88)	36.90 (214.79)	52.72 (305.92)
CPU	+ OpenCL	1.06 (**3.31**)	1.97 (**6.98**)	**2.64** (**9.38**)	**3.87** (**13.77**)	10.33 (**36.76**)	**24.59** (**87.28**)	**34.19** (**122.08**)	**48.90** (**171.13**)
GPU	+ CUDA	**0.07** (0.40)	**0.14** (0.77)	0.21 (1.10)	0.33 (1.70)	0.76 (**2.79**)	1.77 (**6.55**)	2.47 (**9.24**)	3.53 (**12.68**)
GPU	+ OpenCL (Nvidia)	**0.07** (**0.36**)	**0.14** (**0.70**)	**0.20** (**1.01**)	**0.31** (**1.56**)	**0.67** (2.82)	**1.50** (6.76)	**2.07** (9.66)	**2.96** (14.79)
GPU	+ OpenCL (ATI)	0.09 (0.41)	0.16 (0.79)	0.22 (1.08)	0.29 (1.67)	0.79 (3.90)	1.65 (10.52)	2.70 (30.59)	3.24 (886.99)

Table 4.1: Runtime (in seconds) of the different library implementations, computing 1 million local alignments. The c defines the corridor width used for the alignment. The runtimes required for calculating local alignments are shown in parentheses.

linked libraries (DLLs) also called shared objects. Programmers are able to use them by loading the DLL during runtime. This only requires a few lines of code. Each DLL exports functions to calculate the similarity score and the alignment. To optimally use the available hardware, the libraries report a recommended batch size for the score and the alignment calculation.

4.3 Results & Discussion

To evaluate the runtime of the different implementations and hardware platforms, we created eight simulated datasets with read lengths ranging from 36bp to 1000bp. Each data set consists of 1 million reads and the corresponding reference sequences. The length of the reference sequences is determined by $|| G ||=|| R || +c$, where c is the corridor width. A 2% sequencing error is assumed, consisting of mismatches modeled accordingly to Jukes and Cantor (Jukes and Cantor, 1969), insertions, and deletions. Two computers were used for the runtime tests. Both are equipped with two 2.6 GHz Intel Xeon X5650 processors and 32 GB memory. The first computer is equipped with a Nvidia GTX480 card and the second with an ATI Radeon HD 6970 card. OpenSUSE is used as operating system.

First, we compared the performance of the different implementations to the alignment algorithm implemented in the SeqAn (Döring *et al.*, 2008) package. Unfortunately, we were not able to get the banded SW as implemented in SeqAn working. Therefore, we used the semi-global alignment option of the package as a substitute for comparison. Comparisons are based on short read (36bp - 150 bp) and long read (200bp - 1000bp) data. The short read data represents current Illumina technologies, whereas the long read data represents current 454 and Pacific Biosciences technologies. For aligning the short reads we use a corridor size of 15 otherwise of 30.

Table 4.1 shows the runtimes (in seconds) of computing the alignment scores for 1 million reads. As expected, the banded SW alignment shows a near linear runtime behavior for a fixed c whereas the classical SW shows a quadratic runtime behavior as read length increases. For example, aligning 1 million reads of the length 36 takes 10 seconds using the classical SW and 3 seconds using the banded SW. Using a read length of 150 (4 times 36) the runtime of the classical SW increases by a factor of 13 to 132 seconds whereas the banded SW alignment requires only 13 seconds (factor of 4). Table 4.1 also illustrates the influence of the different programming APIs. For SSE we observe an average speedup of 3 compared to the banded SW without SSE. Surprisingly, OpenCL performs slightly better on the CPU than SSE. This demonstrates that the OpenCL compiler uses SSE instruction very efficiently. On a GPU the OpenCL implementation performs slightly better than CUDA for the score computation. However, the differences are marginal. Both GPU implementations outperform SSE by a factor of 14 to 18.

We are only interested in computing the alignment for the genomic region that provides the highest score. Comparing the runtimes of the two banded alignments (Table 4.1) illustrates the benefits of computing the score without the backtracking. Backtracking slows down the computation by a factor of 2 to 3. SSE could not be applied efficiently to the backtracking step as the number of control flow statements makes it difficult to process more than one alignment. Therefore, the SSE library uses the standard banded SW implementation. OpenCL achieves a speedup of 1.5 - 2 compared to the banded SW. Again the graphic card based implementations show a performance boost ($9 - 13$ times faster) compared to the CPU (OpenCL) based implementations. In contrast to the score computation, CUDA outperforms OpenCL when aligning long reads. This is probably due to one of the current shortcomings of the OpenCL API. At the moment there is no way to query the maximal memory allocation size. Thus, we could not use available global memory as efficiently as with CUDA.

Overall the results demonstrate that OpenCL on the CPU as well as on the GPU shows the same performance as SSE and CUDA, respectively. Furthermore, the runtimes demonstrate a clear advantage of using the GPU compared to the CPU for aligning reads and computing scores. So far, we did not take into account the possibility to use more than one core on the CPU.

To investigate how the computation times of our CPU implementations scale with an increasing number of threads we use OpenMP (Dagum and Menon, 1998) for parallelization. Since OpenCL also offers an implicit parallelization mechanism we use two different versions for this comparison. OpenCL CPU1 uses the OpenCL API, whereas OpenCL CPU2 relies on OpenMP to manage the execution on several CPU cores. Due to Intel's Hyper-Threading technologies we could measure the runtime using 1 to 24 threads simultaneously. The resulting speedups compared to a single threaded execution for 1 million 150bp reads are shown in Figure 4.1. Surprisingly, OpenCL CPU2 shows a better performance than OpenCL CPU1. This indicates that the parallelization with multiple cores using OpenMP performs better than using OpenCL directly. Since parallelization is the main focus of the OpenCL API, this is an unexpected result. SSE and OpenCL CPU1 perform equally well. Here, we see an almost linear speedup for up to 12 threads. This corresponds to the number of cores the computer is equipped with. Thus, it appears that our implementations benefit only marginally from Intel's Hyper-Threading.

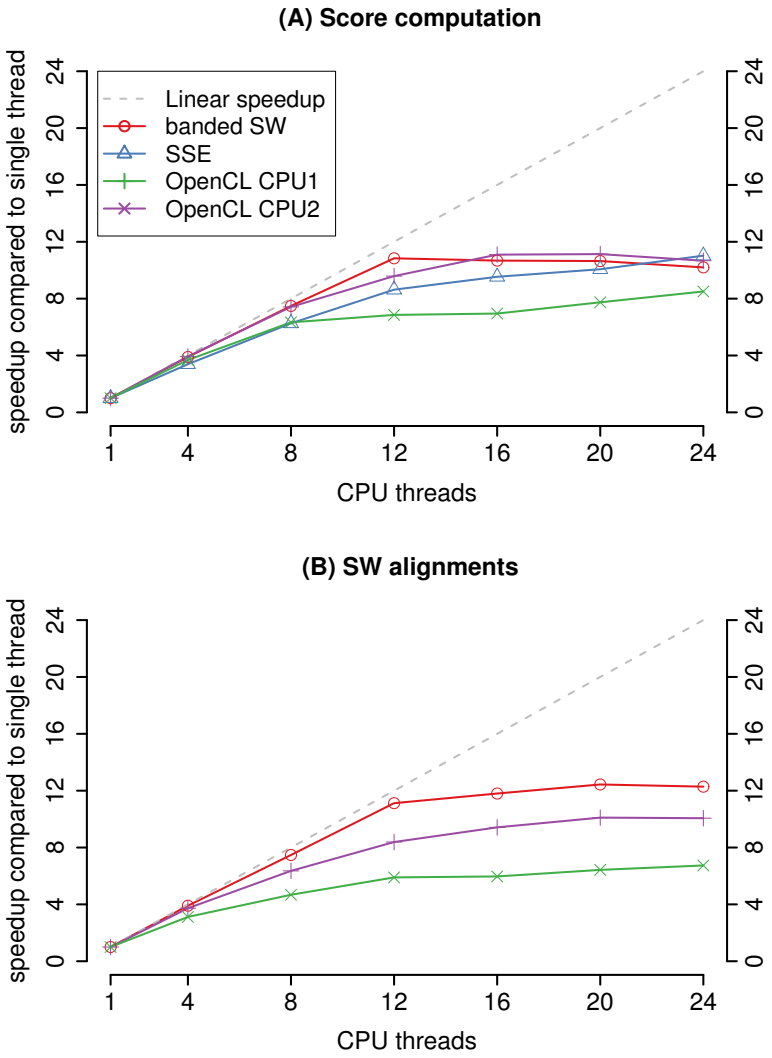

Figure 4.1: Speedup of the CPU based implementations for an increasing number of CPU threads compared to one OpenCL GPU run (dashed line). (A) shows the results of calculating the local alignment scores for 1 million 150bp sequence pairs. (B) shows the result for the same data set calculating the local alignment.

To compare the computational power between a CPU and a GPU we measure the number of scores/alignments computed within the respective runtime of the fastest GPU implementation. For comparison we use an increasing number of threads for the CPU implementations and take OpenCL (Nvidia) as baseline. Figure 4.2 shows the results for 1 million reads of 150bp length. The two Xeon hexa-core processors never achieve the

performance of a single GPU (Nvidia GTX 480), when using the banded algorithm for score computation. Also the SSE and OpenCL CPU2 implementations compute $\sim 10\%$ less local alignment scores than the GPU. For alignment calculation the OpenCL CPU2 version outperforms the GPU based implementation when using more than 12 threads. However, two Xeon X5650 processors compute only $\sim 20\%$ more alignments than a single graphic card. Note that a single Xeon X5650 is more than twice as expensive as a GTX480.

4.4 Conclusion

The increase of read length and number of reads leads to a computational problem when analyzing NGS data. Therefore, the demand for fast, flexible and sensitive processing software for NGS is obvious.

This chapter describes a pairwise local sequence alignment algorithm (Smith-Waterman) adapted to analyze NGS data. We show that our optimizations increase the performance by a factor of 5 to 57 compared to a classical SW implementation.

We demonstrate the benefits of using high performance programming interfaces (SSE, OpenCL, and CUDA) when analyzing NGS data. By incorporating SSE instructions a speedup between 3 to 4 times is observed without additional costs for processing units. Furthermore, usage of wide spread and inexpensive hardware such as graphic cards can substantially reduce runtime. Using a GPU can improve the performance of alignment score computation by a factor of 40 to 55 compared to a single CPU thread. Moreover, our results show that for score computation one Nvidia GTX480 graphic card outperforms two Intel Xeon X5650 processors. A computer equipped with 4 graphic cards and 12 CPU cores computes 16 million alignment scores or 3.3 million alignments for a read length of 150 bp in one second. Score computation for 4×10^8 reads of the length 150 (20-fold coverage of the human genome) requires approximately 75 minutes. Thus, an exhaustive search also on large NGS data can be performed in reasonable time.

Since OpenCL allows writing programs for CPUs as well as for GPUs we investigated its applicability for sequence alignments. Our results show that OpenCL outperforms SSE for reads longer than 72bp on the CPU. On the GPU OpenCL outperforms CUDA for reads smaller than 200bp. Interestingly OpenCL could not handle multiple CPU

threads as efficiently as OpenMP. However, we are confident that this will be resolved in the near future.

The key outcome of this work is MASon, a C++ library that is capable of processing millions of short (36bp - 1500bp) sequence alignments efficiently. MASon includes SSE and OpenCL based implementations for CPUs as well as GPU based implementations using CUDA and OpenCL. All implementations are encapsulated in dynamic linked libraries and can therefore be easily integrated into existing or upcoming applications for NGS. By using our library system every developer is able to write applications that optimally utilize modern hardware, ranging from desktop computers to high-end cluster systems. Future work will include extending the libraries with a global alignment algorithm.

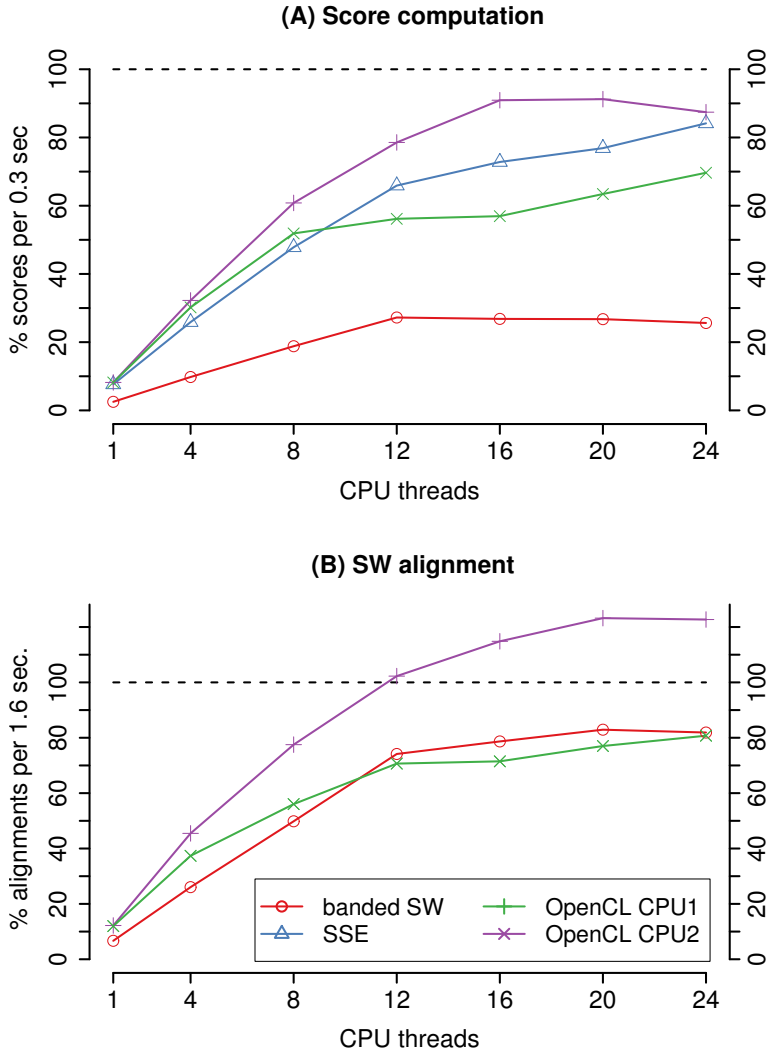

Figure 4.2: Percentage of score (A) and alignment (B) computations compared to OpenCL (Nvidia GPU) for 1 million 150 bp long sequence pairs.

Chapter 5

On the accuracy of MiSeq and Ion Torrent sequencing technologies

5.1 Introduction

In the previous chapters 3 and 4 we discussed methods to align Next Generation Sequencing (NGS) reads to a reference genome. To obtain a reliable result from a NGS related experiment, one also has to take into account the specific limitations and errors that come along with each NGS technologies. As some NGS technologies are better suited to answer particular questions than others (Glenn, 2011) the choice of the NGS technology becomes a key task in experimental design. In this chapter, we will give insights into the performance (e.g. number of reads) and error structure for two benchtop sequencing technologies. Although we are focusing on two specific technologies, the methods and algorithms applied here can be used for every sequencing technology.

Life Technologies was the first to release a benchtop sequencing platform named Ion Torrent (Rothberg *et al.*, 2011). Illumina followed with their sequencing platform named MiSeq (Glenn, 2011). Both technologies are a cost efficient alternative to sequence small DNA samples within hours.

As of August 2011, Life Technologies and Illumina each benchmarked the performance of their sequencing platform based on a re-sequencing of the genome of *Escherichia coli* Dh10b. Life Technologies claimed the outstanding performance of Ion Torrent in terms of uniform coverage of the reference genome and low number of consensus mismatches (Life Technogies Corp., 2011b) when compared to MiSeq. Interestingly, Illumina claimed

in their comparison of the two technologies also the outstanding performance of MiSeq in terms of a low read error (Illumina Inc., 2011b). However, Illumina assessed the performance by counting insertions and deletions without focusing on mismatches. Thus, we now have two comparisons of the sequencing technologies at hand. It is somehow unfortunate that their result, although the data was the same, cannot be compared due to the companies differing quality criteria.

Here we investigate the accuracy of both technologies using an unified scheme of quality assessment. First, we analyze the performance focusing on the individual reads. We first measure the sequencing error and its distribution across the reads, as well as possible sequencing error biases related to each of the four nucleotides. We define sequencing error as the percentage of differences (mismatches, insertions and deletions) between the read and the reference sequence. Second, we compare the performance of both technologies focusing on the genomic level. Specifically, we assess the per base coverage distribution along the reference genome as well as the performance to obtain the biological signal such as the distribution of variant sites. In this context, we define variant sites as positions that differ between the reference genome and the sequenced genome.

5.2 Methods

5.2.1 Data

To assess the performance of MiSeq and Ion Torrent, we analyzed publicly available sequencing data of *Escherichia coli* strain Dh10b. We obtained 5 Ion Torrent data sets (C22-169, GAT-541, B13-328, B15-410, B14) sequenced by Life Technologies Inc. (Life Technogies Corp., 2011a). In addition, we obtained one Ion Torrent data set sequenced in Münster, Germany (MU). For simplicity we will refer to all Ion Torrent data sets using the prefix "Ion_". Currently, Life Technologies offers more data sets for Dh10b. We selected the subset such that 3 chip technologies (314, 316, 318) are represented. B14 (B14) was generated with Chip 314. B13-328 (B13), B15-410 (B15) and MU are based on Chip 316 and C22-169 (C22), GAT-541 (GAT) were produced on Chip 318. The comparison of this data should, therefore, provide a comprehensive overview of the general characteristics of Ion Torrent, and should highlight advantages as well as

current limitations of this technology. For the MiSeq technology, we obtained one data set for *Escherichia coli* strain Dh10b sequenced by Illumina Inc. (Miseq-110721-R1, Miseq-110721-R2) (Illumina Inc., 2011a).

5.2.2 Mapping of reads

We used NextGenMap (see chapter 3) to map the reads to the reference genome of *Escherichia coli* Dh10b. NextGenMap uses a seed word approach to identify putative mapping regions. We chose a word length of 12bp throughout all the mapping runs. For each of the putative mapping regions NextGenMap computes semi-global alignment scores for the entire read. For the best scoring region then a semi-global alignment (Gusfield, 1997) is computed. This guarantees that the read is aligned over its entire length. When a read has more than one best scoring region on the genome we cannot place it unambiguously. To take the resulting ambiguity in the placement of this read into account in the downstream analysis, e.g. when computing the per base coverage or the sequencing error, we assigned a weight of one divided by the number of best mapping positions. Afterwards we computed the per read sequencing error as the number of mismatches, insertions and deletions divided by the alignment length. The reads that were mapped to the minus strand were reverse complemented to preserve the assignment of sequencing order for the position wise computation of the sequencing error. The position wise sequencing error was computed based on all alignments summing up the number of mismatches, insertions and deletions per position in the alignment.

5.2.3 Identification of variants

The identification of variant sites is an important procedure when sequencing a genome. Variant sites are positions in the genome of the sequenced individual or clone that differ from the reference genome. To assess the influence of the technologies on the detection of sequence variants, we used the example of comparing two *Escherichia coli* strains, Dh10b and W3110. The latter strain was used as a reference sequence. For the analysis we first assessed the set of true sequence variants distinguishing the two strains. To this end, we aligned the genome sequence of W3110 against the genome of Dh10b. Mauve 1.3.1 (Darling *et al.*, 2010) was used with the default parameters (Table 5.1). Any sequence difference observed in the genomic alignment was taken as the truth.

Parameter	Value
Default seed weight	yes
Use seed families	no
Determine LCBs	yes
Assume collinear genomes	no
Full Alignment	yes
Iterative refinement	yes
Sum of pairs LCB scoring	yes
Min LCB weight	yes
Scoring matrix	HOXD

Table 5.1: Parameters used for genomic alignment.

In the next step, we mapped the reads from the data sets of Dh10b to the genome of W3110. Mapped reads that have a higher difference than 10% were discarded. Variant detection was carried out using samtools mpileup with recommended parameters, followed by bcftools and vcftools (Li *et al.*, 2009). The minimum coverage was set to 5. Note, that variant calling was performed using only the uniquely mapped reads. The obtained variants were rejected if their quality value was < 20 (Q20) (Li *et al.*, 2008) or if the variant frequency (number of reads that support the variant at a specific position) was below 90%. Variant qualities are phred-scaled quality scores that represent the confidence of the alternative allele. We computed the variant frequency based on all reads that support the variant divided by the number of reads that overlap the variant positions. The variants that survived the filtering thresholds were then taken as experimentally derived candidates. We next compared our experimentally candidate variants to the variants obtained from the whole genome alignment and computed the true positive, false positive and false negative rates. We define false positives as variants that are not in the set of true differences. As true positive we consider positions that are part of the true differences. False negatives are variants that were not found but are present in the true set of variants.

5.3 Results

5.3.1 Comparison on the raw read level

To investigate the performance of the two sequencing technologies we first focused on the read sequences. Table 5.2 shows the read numbers, lengths and redundancies per data set. In this context, we define redundancy as the number of reads that have the same nucleotide sequence and length. As an example, given 10 reads and 5 are copies of one read, than the redundancy is 50%. Both MiSeq data sets have a read length of 150bp (Table 5.2), which is the current maximum read length of Illumina (Glenn, 2011). Ion Torrent reads displays a large variation of read lengths ranging from 3bp up to 396bp. The average redundancy of reads across all Ion Torrent data sets is substantially (0.35%) than for the MiSeq data (17.56%). Given our definition of redundancy, this may be partially an effect of the varying read lengths in the data sets of Ion Torrent. Despite their higher redundancy, MiSeq still obtain in absolute numbers the higher number of unique reads due to its substantially higher read output (2 × 8.5mil).

Data sets	# Reads	Mean length	# Bases pairs	Red. (%)	Rel.date	Chip
MiSeq R1	**8,584,754**	150.0	**1,296,297,854**	19.75	8-2011	HiSeq
MiSeq R2	**8,584,754**	150.0	**1,296,297,854**	15.36	8-2011	HiSeq
Ion_GAT	5,255,430	228.1	1,203,970,524	0.58	10-2011	318
Ion_C22	6,479,267	**240.0**	1,561,441,854	0.27	12-2011	318
Ion_B15	2,761,903	239.5	664,131,578	0.37	10-2011	316
Ion_B13	1,687,490	100.3	175,570,901	0.78	6-2011	316
Ion_MU	3,055,192	212.8	653,210,758	**0.04**	9-2011	316
Ion_B14	350,109	222.0	78,073,794	**0.04**	8-2011	314

Table 5.2: Read statistic from the individual data sets. For MiSeq we obtained only one data set of paired end reads marked by R1, R2. The data sets with the prefix Ion_ were sequenced by three different chip generation from Ion Torrent. We define redundant reads (red.) as the percentage of reads that have an identical nucleotide sequence and read length.

The quality values for the various sequencing runs are summarized in Figure 5.1.

Typically the higher the quality value the higher is the probability that the nucleotide is correctly called (Cock *et al.*, 2010). Both technologies show an increase in low quality values towards the ends of the reads. The quality values of the two technologies are not directly comparable as they both use different software to compute the values (Glenn, 2011; Loman *et al.*, 2012). Note, two sites each in the two MiSeq distributions represent obvious distortions in the distribution of the quality scores (R1: Pos 34 and 99 ; R2: 29 and 79). Presumably the increase of the low quality values is due to a technical problem during sequencing.

5.3.2 Sequencing error estimation

We aligned the reads to the *Escherichia coli* Dh10b reference genome using a semi-global alignment (Gusfield, 1997), i.e., we align the whole read to the reference. Table 5.3 summarizes the mapping statistics for uniquely mapped reads. As we are interested in the sequencing error of the machines, we investigated the sequencing error distribution across all mapped reads. NextGenMap could uniquely map 93.04% and 93.00% for MiSeq R1 and MiSeq R2, respectively. For Ion Torrent, NextGenMap could uniquely map between 93.53% to 92.64% of the reads.

First, we computed the overall sequencing error per technology based on all mapped reads. We measured the sequencing error per position over all the alignments based on the number of observed mismatches, insertions and deletions at each site. For comparing the sequencing error of each technology we chose the mean and the median position wise sequencing error across all alignments. Table 5.4 summarizes the results. The Ion_B14 data show the highest mean sequencing error of 20.68%, while MiSeq R1 has the lowest error of 1.94%. The median sequencing error across the alignment positions reveals a slightly different picture (Table 5.4). The data sets of Ion_B13 and Ion_B15 have the lowest error rates of 1.36% and 1.30% respectively. Still, the Ion_B14 shows the largest median sequencing error (10.68%) of all data sets .

Next we assessed the sequencing error for those reads that show a sequencing error $\leq$ 10%. The percentages of remaining reads are shown in Table 5.3. Not surprisingly, the mean and median sequencing error is reduced. For the mean sequencing error MiSeq R1 shows with 0.06% the lowest sequencing error. Ion_MU shows the highest mean sequencing error (3.23%) followed by Ion_B14 (2.89%). When comparing the median

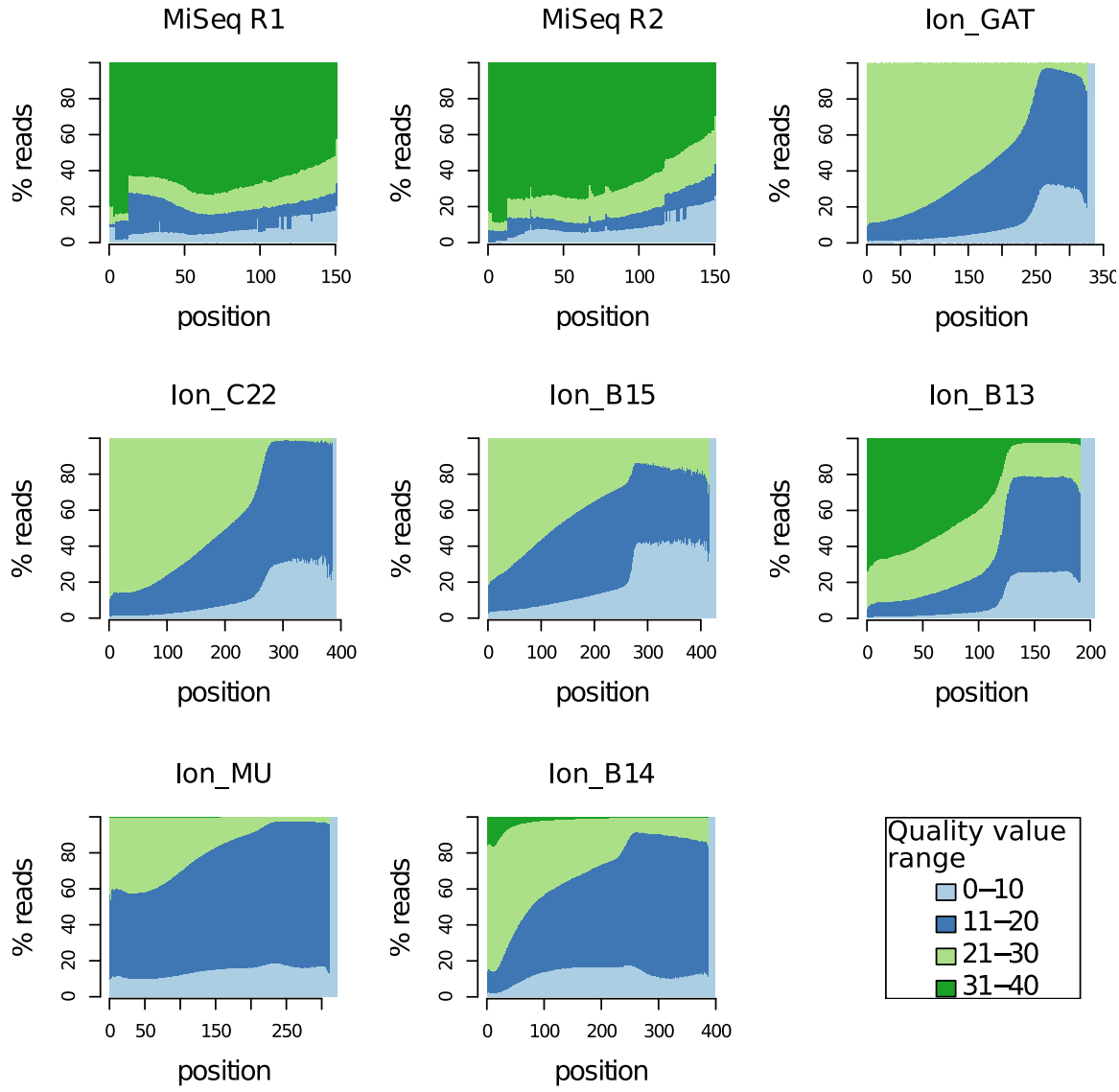

Figure 5.1: Per base quality values quantized into 4 categories. The percentage of reads was normalized per read position.

sequencing error, all data sets apart from Ion_B14 and Ion_MU show a median error $\leq 1.00\%$. MiSeq R1 has again the lowest median sequencing error of 0.32%.

Figure 5.2 shows the position wise sequencing error accross the alignments of reads having a sequencing error $\leq 10\%$. The sequencing error increases to the end of the alignments. When we analyse the different types of errors, i.e. mismatches, insertions and deletions, we observe for MiSeq an increased mismatch rate compared to the insertion

Data sets	Uniq. mapped (%) All	Uniq. mapped (%) With thresh. (%)	Not covered pos.	Mean cov.
MiSeq R1	93.04	80.74	70	120.01
MiSeq R2	93.00	78.77	118	117.04
Ion_GAT	93.37	82.90	**67**	115.00
Ion_C22	**93.53**	84.42	72	**150.71**
Ion_B15	93.19	**88.09**	90	67.93
Ion_B13	92.93	86.60	111	18.68
Ion_MU	92.64	60.52	143	44.55
Ion_B14	92.72	59.81	86,584	4.92

Table 5.3: Mapping statistics of the 8 data sets from *Escherichia coli* Dh10b. The second column depicts the percentage of uniquely mapped reads. The third displays the % of uniquely mapped reads that have more then 10% differences (wT) respectively. All sequencing data sets share 30 positions that are not covered. Those are the first 30 position of the reference genome and might be due to a bias introduced in the used programs.

and deletion rates across all positions. This is a typical sequencing error pattern for Illumina technologies (Glenn, 2011; Suzuki *et al.*, 2011). For MiSeq R1 97.97% of all errors are due to mismatches. For Miseq R2 we observe that 98.09 % of all errors are due to mismatches. The situation is completely different for the Ion Torrent data. For the data set Ion_B14 insertions and deletions make up for 93.78% of the sequencing errors within the first 222bp (average read length). Whereas above 222bp the fraction of mismatches for a given sequencing error increases to 36.86%. The high insertion/ deletion rate of 93.78% for the first 222bp can be explained by the sequencing technology (Glenn, 2011). This is, Ion Torrent measures the number of inserted nucleotides based on the release of H^+, which makes it hard to determine the exact number of inserted nucleotides. However, we speculate that the increased mismatch rate (36.86%) towards the end of the alignments is caused by the pairwise semi-global sequence alignment. Since the cost of a mismatch is typically lower than the insertion or deletion costs, alignment tools tend to favor mismatches over insertions/deletions towards the end of the sequences. The number of alignments decrease above 222bp (average read length), given an error at the

Data sets	Mean seq error (%)			Median seq error (%)		
	(0%)	(90%)	(90%)_150	(0%)	(90%)	(90%)_150
MiSeq R1	**1.94**	**0.55**	**0.55**	1.47	**0.32**	**0.32**
MiSeq R2	2.72	0.80	0.80	2.15	0.46	0.46
Ion_GAT	3.37	1.84	0.61	1.94	0.92	0.52
Ion_C22	3.17	2.01	0.56	1.66	0.92	0.46
Ion_B15	2.10	1.55	0.55	**1.30**	0.87	0.49
Ion_B13	3.27	1.63	1.63	1.36	0.46	0.46
Ion_MU	10.59	3.23	1.11	4.73	1.49	0.85
Ion_B14	20.68	2.89	1.00	10.68	1.75	1.00

Table 5.4: Mean and median sequencing errors per position over all mappable reads for different similarity cutoffs. The columns "_150" depicts the resulting mean sequencing error across all technologies with the alignment length of MiSeq (150bp).

end of the read it is more likely to be aligned with a mismatch than an insertion or deletion. This would lead to an increase in the percentage of mismatches as we observe it. The same holds for the other Ion Torrent data sets. In summary, we observe an increased insertion/ deletion rate compared to mismatch rate in all Ion Torrent data sets. However, towards the end of sequence reads an increased percentage of mismatches is observed.

Since the Ion Torrent technology yields longer reads (average: 222bp) compared to MiSeq (average: 150bp) we compared the sequencing error only for the first 150 positions in the alignment. For Ion_B14 the mean sequencing error reduces from 2.89% to 1.00%. Three Ion Torrent data sets (Ion_B15, Ion_GAT, Ion_C22) and Miseq R1 have a mean sequencing error $\leq 0.61\%$. When measuring the median sequencing error all technologies have a sequencing error $\leq 1.00\%$.

Next, we computed the nucleotide frequencies given a sequencing error to assess a possible bias. The frequencies of the four nucleotides in the reference genome of *Escherichia coli* Dh10b are almost the same (A= 24.76% ,C= 25.21%, G= 25.33% ,T=24.70%). Given this and given the assumption that the coverage across the genome is more or less uniform we would expect that the observed nucleotide frequencies given a sequencing

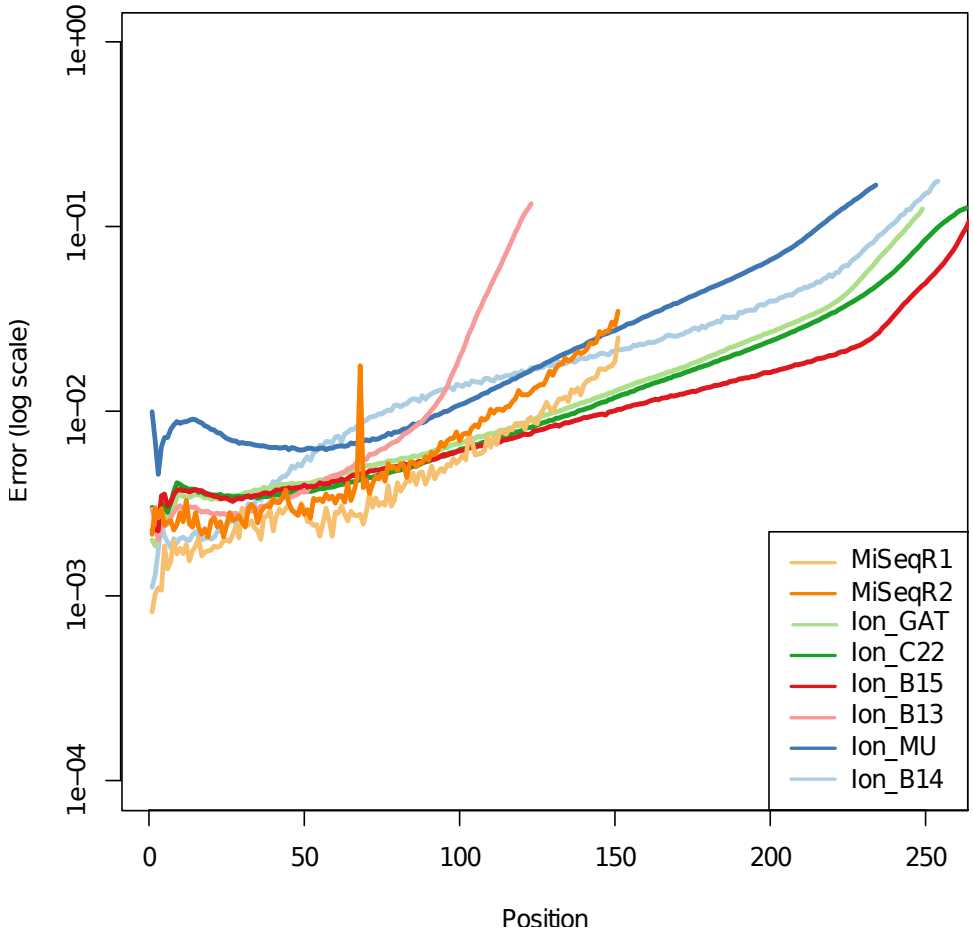

Figure 5.2: The per base sequencing error of all 8 data sets. The sequencing error was computed based on all reads that could be aligned given a maximum of 10% errors in the alignment.

error are also uniformly distributed. First, we investigated the sequencing error structure based on all mappable reads without a threshold. Figure 5.3 shows the nucleotide distribution of the mapped reads given a sequencing error normalized by the number of errors per alignment position.

We measure a decreased probability of mismatches given a G (16.07%) in the reference compared to A, C, T (25.16%, 26.95%, 22.62%) for Miseq R1 and similar values for MiSeq R2. For MiSeq R1 we identified one site (position 34) with an increased mismatch rate (43.12%) given a G in the reference. Note that the positions corresponds to the observed positions of fluctuation of quality values for R1 (Figure 5.1). We observe two slightly

increased position of indels at position 122 and 135 having 20.71% (insertion: 5.95%, deletion: 14.76%) and 18.60% (insertion: 5.20% deletion: 13.40%) of indels respectively compared to an average rate of 15.89% (insertion: 5.76%, deletion: 10.12%) of indels across all alignments. For MiSeq R2 we observe two sites at position 29 (39.47%) and 79 (34.72%) showing an increased mismatch rate given a G in the reference. Note that this corresponds to the observed fluctuation of quality values for R2 (Figure 5.1). At position 68 we notice an increased sequencing error due to mismatches from 83.88% on average to 90.24% given an A, C and T in the reference genome.

For the Ion Torrent data sets we measure an increased average insertion rate of an A (12.95%) compared to the other types of insertions, across all Ion Torrent data sets. Ion_B14 has an increased percentage of having an insertion of an A (14.06%) compared to inserting other nucleotides across the alignment (insertion: C (7.59%), G (6.91%), T(8.76%)).

When we repeat the entire analysis focusing only on reads with an sequencing error $\leq 10\%$ the overall picture does not change. Figure 5.4 shows the nucleotide distribution of the mapped reads given a sequencing error. For MiSeq R1 and R2 the majority of the increased sequencing error sites are removed, apart from position 68 MiSeq R2, that shows an increase of mismatches from 90.24% to 99.92% given a sequencing error. For Ion Torrent, we measure on average accross all data sets, an increase in the insertion rate of an A from 12.95% to 14.51%. The second most abundant sequencing error type is a deletion given an C (12.94%) in the reference.

Next we investigate if the position of the reads on the sample carrier in the sequencing machine is associated with an increase in sequencing error. We extracted the location of a read on the sample carrier based on its read name. A density plot is computed based on the number of reads distributed over the sequencing machine. The results are shown in Figure 5.5. Here we compare the location of reads with a sequencing error $\leq 10\%$ to those reads showing a sequencing error $\geq 30\%$. Reads with a sequencing error $\geq 30\%$ are usually rejected from the analysis as they are believed to origin from contamination or other reasons. However, if the source of dissimilarity is related to library preparation, e.g. contamination and amplification artifacts, those occurrence would be uniformly distributed over the sample carrier. For Ion Torrent, we observe that the majority of reads with a sequencing error $\geq 30\%$ originated in the upper right corner. For MiSeq, we observe those reads having a sequencing error of $\geq 30\%$ at the border of the sample

carrier. Whereas those that show a sequencing error $\leq 10\%$ are in the middle of the sample carrier.

5.3.3 Comparison on the genomic level

5.3.3.1 Coverage distribution

Achieving an uniform coverage of the sequenced genome is important for nearly every NGS experiment to determine the e.g. site frequency spectrum or transcription rate. We computed the per base coverage accross the reference genome to investigate the ability of each technology to achieve an evenly covered genome. The per base coverage is computed by the number of overlapping reads per position of the reference genome. As shown in Table 5.3 there are 86,584 positions for Ion_B14 not covered by reads. However, we did not observe contiguous not covered regions, apart from the first 30 positions. As they occur in every sequencing data set, they might be an artifact of the programs. Since the number of reads vary between 350,109 (Ion_B14) and 8.5 mil (MiSeq R1) for the different data sets we adjusted the per base coverage using the Z transformation (Ragazzini and Zadeh, 1952). This shifts the mean value to 0 and normalize the standard deviation to 1. Figure 5.6 shows the average coverage per 1000bp adjusted by the overall depth of sequencing. All sequencing technologies show a relatively even coverage across the genome.

5.3.3.2 Identification of variants

Often sites that differ between the reference genome and the sequenced genome are of interest (Nordborg *et al.*, 2005; Mackay *et al.*, 2012). The accuracy of benchtop sequencers to reveal such variant sites is an important information. A genomic alignment between the *Escherichia coli* strains Dh10b and a close relative W3110 revealed 179 mismatch positions between W3110 and Dh10b. Those were taken to assess the suitability of the sequence performance of the two NGS technologies for the identification of the true sequence variants.

First, we mapped the Dh10b reads from each data set to the *Escherichia coli* strain W3110. Table 5.5 shows the percentage of uniquely mapping reads with a sequencing

error $\leq 10\%$. Based on these reads we derived the variant positions using samtools (Li *et al.*, 2009).

Data sets	Uniq. mapped wt (%)	detected	True positive	False positive	missing SNPs
MiSeq R1	86.31	165	153	12	26
MiSeq R2	88.98	**166**	**154**	12	**25**
Ion_GAT	86.29	162	147	15	32
Ion_C22	87.59	159	142	17	32
Ion_B15	**91.61**	153	147	6	32
Ion_B13	90.42	144	144	**0**	35
Ion_MU	62.86	150	146	4	33
Ion_B14	62.37	104	103	1	76

Table 5.5: Variant discovery based on *Escherichia coli* W3111. The reference set of true variants was computed based on a genomic alignment between *Escherichia coli* Dh10b and *Escherichia coli* W3111. In total 179 variants could be discovered, based on the genomic alignment. We define false positive as differences that are not in the set of the 179 true variants. True positive are those differences that have also been found in the genomic alignment. False negatives are those differences that were not found by the sequencing data sets but appeared present in the genomic alignment. We did not list the true negative since those are all remaining positions of the *Escherichia coli* genome.

None of the data sets identify all true variants. The MiSeq data sets R1 and R2 showed both a true positive number of 153 and 154 by identifying 165 and 166 variant sites respectively. When combining both MiSeq data sets we identified 168 variants resulting in a true positive number of 155. Ion_B14 discovered 104 variant sites with the lowest true positive number of 103. This is probably due to the low coverage of this particular data set (mean coverage of 4.9). Ion_B15 has 147 true positives. When combining Ion_B15, Ion_B13 and Ion_GAT the true positives increased to 172 compared to both MiSeq data sets having 155 true positives. Combining the two MiSeq and Ion_B15 data resulted in 170 true positive variant sites out of 179 differences.

5.4 Discussion

Here we assess the performance of MiSeq and Ion Torrent based on previously published data sets. We measure the total sequencing error as the percentage of observed differences (mismatches, insertions and deletions) comparing the read with the aligned region in the reference genome of *Escherichia coli* Dh10b. In agreement to the Illumina application note (Illumina Inc., 2011b) and Loman *et al.* (2012), we observe that MiSeq has a lower total sequencing error compared to Ion Torrent. In our case, we measure an increased error of Ion_B14 (2.89%) compared to the MiSeq (R1: 0.55% and R2: 0.80%) data. Only when measuring the median sequencing error across all mappable reads we observe that Ion_B15 (1.30%) and Ion_B13 (1.36%) have a lower sequencing error compared to MiSeq R1 (1.47%) and R2 (2.15%).

In recent years multiple paper appeared that discussed the sequencing error across different NGS platforms (Harismendy *et al.*, 2009; Suzuki *et al.*, 2011; Loman *et al.*, 2012). To contribute to this field, we have investigated how the measurement of mismatches, insertions and deletions influence the total sequencing error. For the same data set (Ion_B14) processed with the same method, we measured a sequencing error ranging from 20.68% to 1.75%. This is a 12 fold difference in the sequencing error by measuring one time the mean sequencing error over all mapped reads (20.68%) and one time the median sequencing error over all mapped reads having a sequencing error $\leq$ 10% (1.75%). Clearly, this is not an useful comparison, however, it shows how reliable a measurement of sequencing error might be. Furthermore, the measurement of the sequencing error might be influenced by the chosen mapping method. For example, Suzuki *et al.* (2011) measured the sequencing error based on the mapping results obtained from Bowtie (Langmead *et al.*, 2009) . Using the default options for Bowtie maximum of 3 mismatches are allowed. Given the average read length of Ion_B14 this would result in alignments with a maximum of 1.35 % errors, ten times less what was used here. In addition, Bowtie is not able to handle insertion or deletions (Langmead *et al.*, 2009; Li and Homer, 2010). Both would result in a reduced sequencing error for Ion Torrent, in the same time lowering the number of mappable reads. For our study we did not apply quality trimming as we were interested in the sequencing error characteristic of each technology. We used a semi global alignment approach to align the full read length. Using a local alignment method in combination with quality trimming will lead to a reduction of measured sequencing error. We further speculate that the total sequencing

error depends on the scoring matrix, used to map the reads to the reference genome. However, we did not investigate this here. Comparable to Suzuki *et al.* (2011) we did not use the quality values for the analysis and we did not take the paired end information for MiSeq into account.

Loman *et al.* (2012) published a comparison of currently available benchtop sequencing technologies focusing mainly on insertion and deletion rates of the sequenced reads. In contrast to the paper of Loman *et al.* (2012), we investigated the accuracy of Ion Torrent based on multiple data sets published between June and December 2011. Thus, we could investigate the sequencing accuracy of different chip technologies (314, 316 and 318) for *Escherichia coli* Dh10b. In our study the data sets Ion_B13 and Ion_B14 were from chip generation 314 and 316 whereas Ion_C22 and Ion_GAT were produced using chip 318. We observe (Table 5.2) an improvement of the newer chip (318) in terms of number of reads, mean coverage, not covered positions and a decreased sequencing error.

It is also for the first time that the nucleotide frequency given a sequencing error across all alignments was measured and investigated. Since it reveals biases given a nucleotide in the reference we speculate that this is an important information for every SNP analysis. For MiSeq, we observed a decreased frequency of mismatches given a G (16.07%) in the reference compared to the other nucleotides A (25.16%), C (26.95%) and T (22.62%) or indels (9.20%). In case of Ion Torrent, we measured an increased average frequency of having an insertion of an A (14.51%) across the available data sets. A deletion given a C (12.94%) in the reference has the second highest sequencing error frequency on average across all Ion Torrent data sets.

To gain additional insights into the sequencing process we measured the distribution of reads across the sample carrier in the sequencing machine. Here we observe a clustering of reads having a sequencing error $\geq 30\%$ to the right top of the sequencing chip of Ion Torrent. For MiSeq we observe reads having a sequencing error $\geq 30\%$ on the border of the sequencing slide and at one line crossing the slide. This cannot be the result of a contamination or error in the library preparation as those would result in a more uniform distribution. One possible explanation for this observation would be that the distribution of nucleotides on the sample carrier is often not uniform and therefore leads to the incorporation of wrong nucleotides.

The performance comparison to identify sequence variants across the genome was carried out based on a genomic alignment between *Escherichia coli* Dh10b and *Escherichia*

coli W3111. In our analysis, non of the used sequencing data sets could recover all true variants. MiSeq revealed the highest true positive number of 156. However, a combination of MiSeq and Ion Torrent data revealed an increase of true positive variant sites of 157. This shows that a combination of both technologies might be the best way to obtain new insights. However, due to the combination one has to note that the overall coverage is also increased. This contribute to the increase of true positives.

In agreement with Glenn (2011) we suggest to define and use a set of standard method or even establish a standardized workflows for measuring sequencing error. To promote this, we combined all methods used in this analysis together in a program called *MoNTY* (Measurements fOr Ngs TechnologY). This requires a mapped read file and the used reference genome for mapping. In addition to a standard pipeline, we suggest sequencing always the same defined set of organisms and strains. Only the identical input material in combination with the usage of the same methods to analyse the data allows for an objective comparison of each sequencing technology.

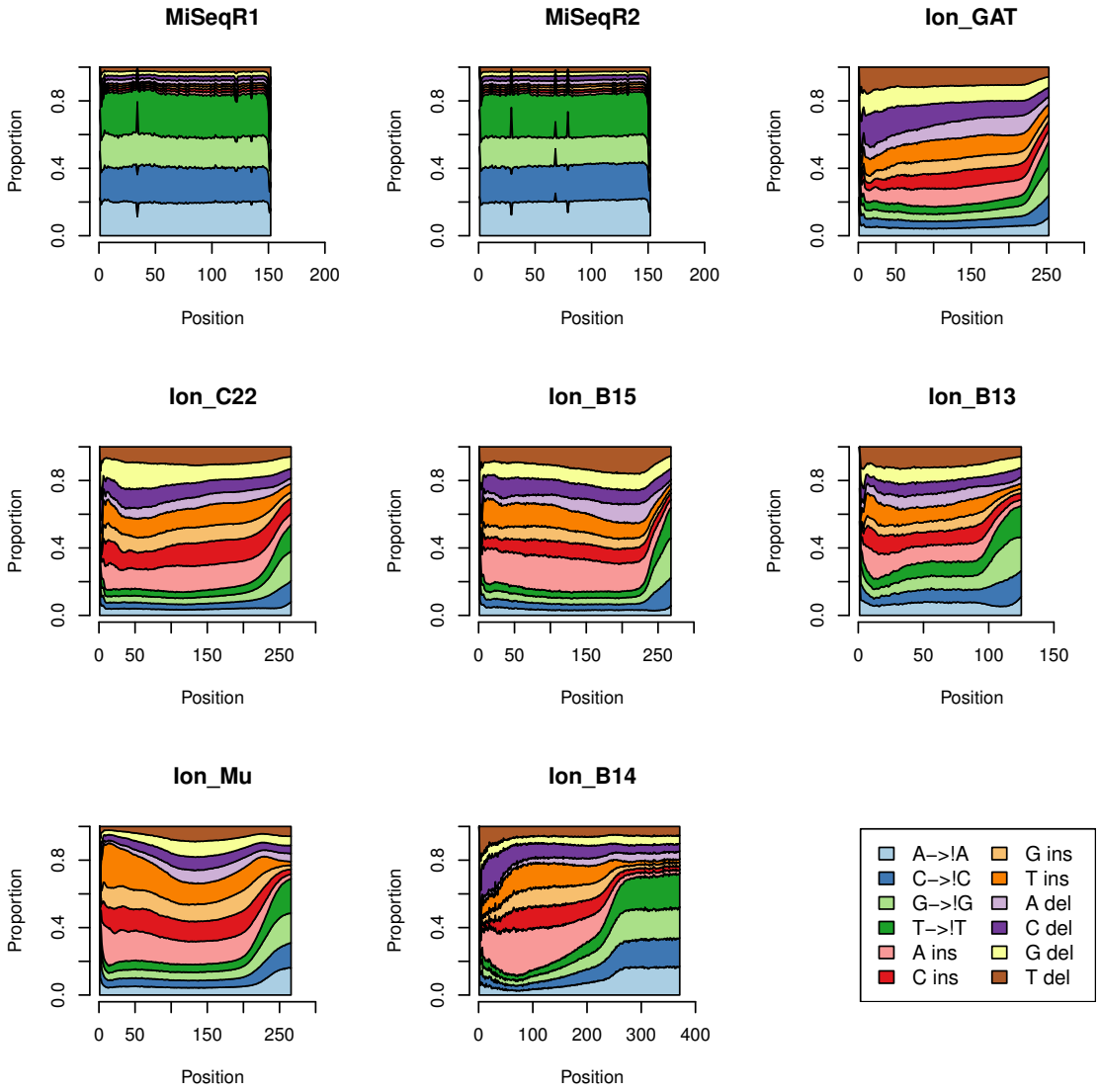

Figure 5.3: Nucleotide frequencies given a sequencing error in the alignment for all mapped reads. The types of differences are summarized into 12 categories which includes mismatches (mutation from A (ref) to a non A: A->!A), insertion (an insertion of an A: A ins) and deletions (a deletion of an A: A del). As an example, given a set of reads from which 25% show a mismatch of an A (reference) to a C, the figure would show a light blue (A->!A) peak on the first position. This indicates that the set of reads seem to have a bias towards this mutation at position 1. In the following, we refer always to the percentage given a sequencing error.

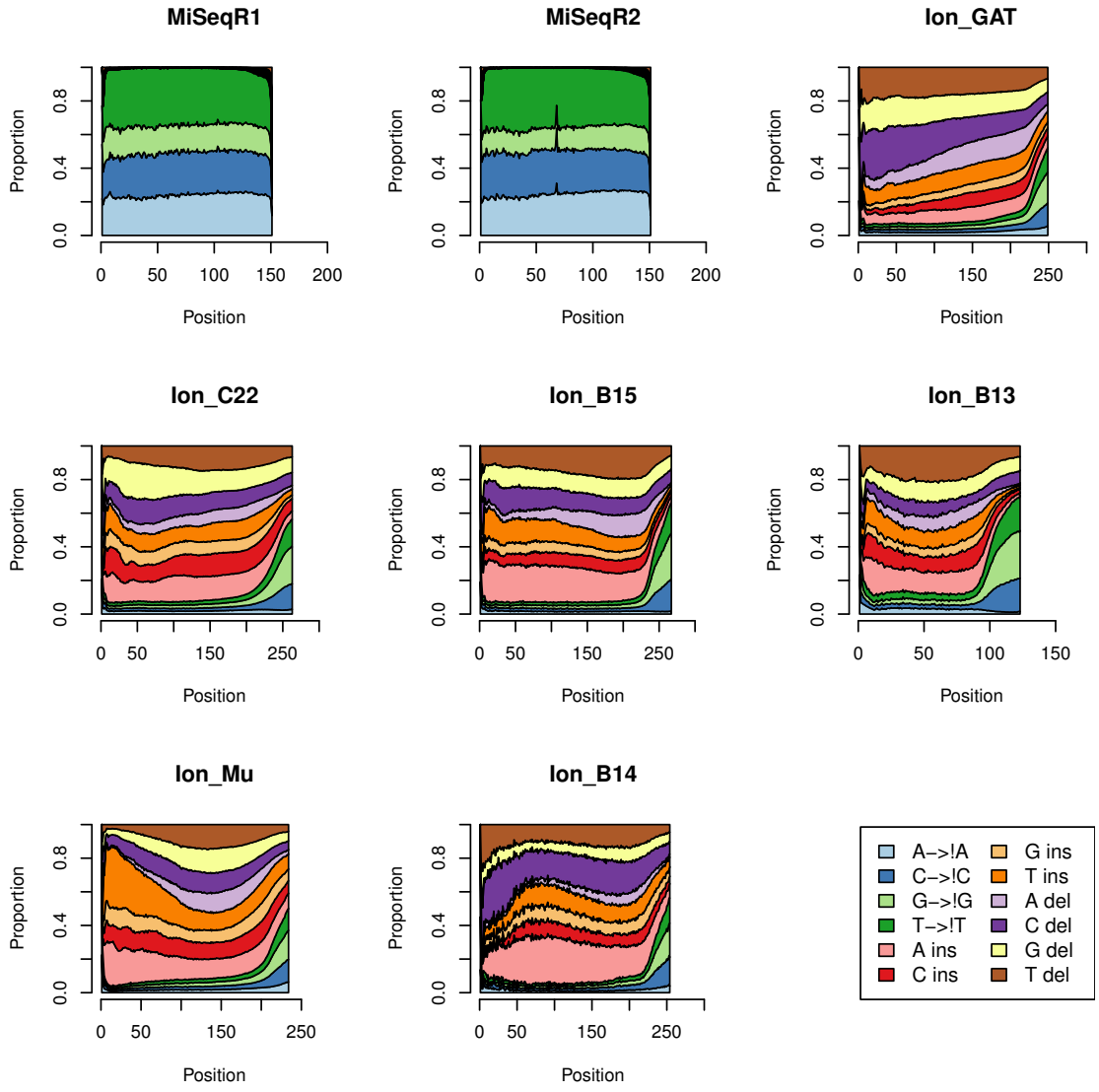

Figure 5.4: Nucleotide frequencies given a difference in the alignment for a read set of maximum of 10% errors in the alignment.

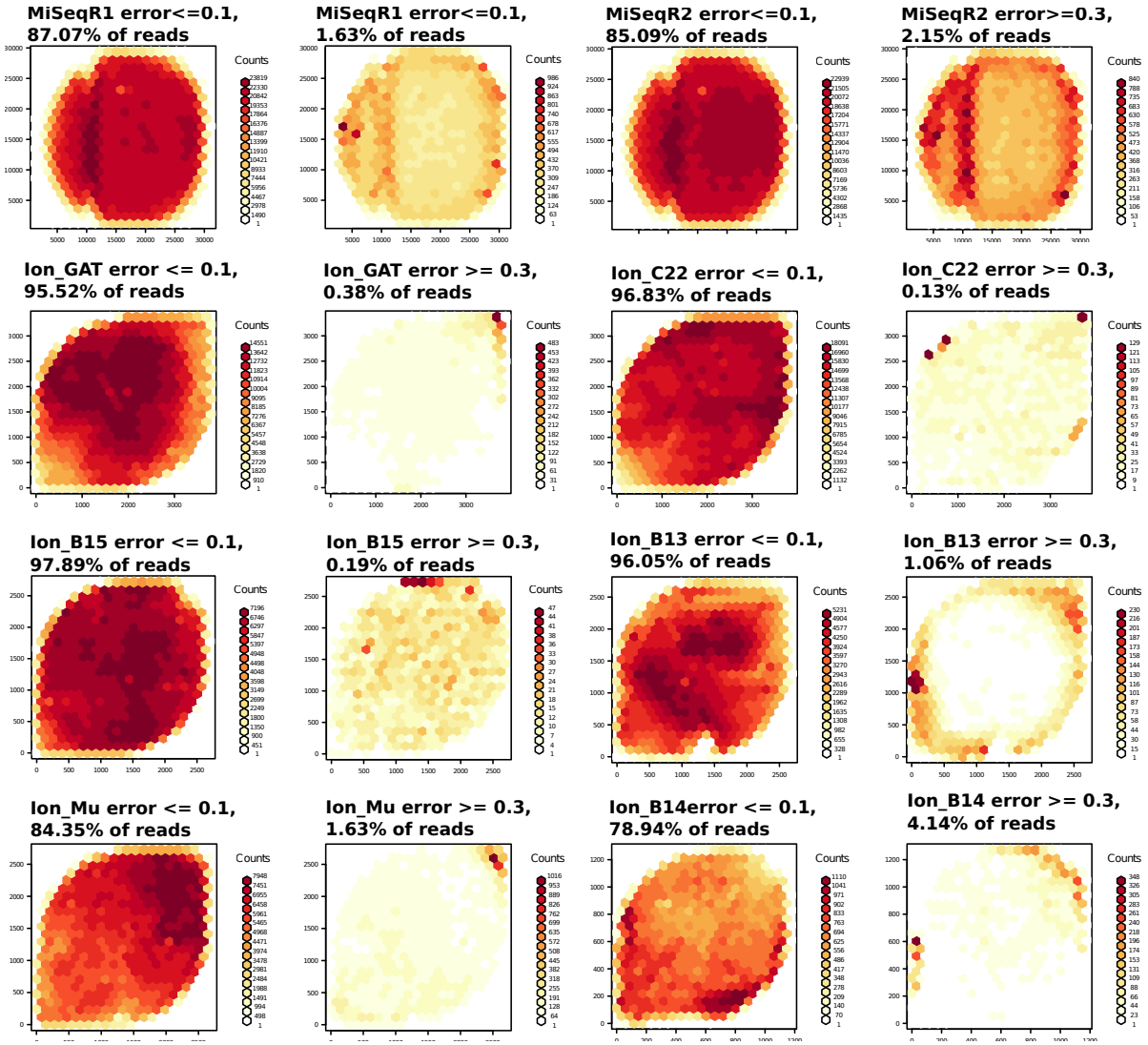

Figure 5.5: Density plot of the read location on the sample carrier in the sequencing machine: The density from white (no) to dark red (high density) of reads given their errors compared to the genomic location. Interestingly we observe a clustering in the upper right corner for the Ion Torrent data sets of reads that have a sequencing error of $\geq 30\%$. For MiSeq, we observe a clustering of reads with a sequencing error of $\geq 30\%$ on the border of the sequencing area as well as at one line across the sequencing slide.

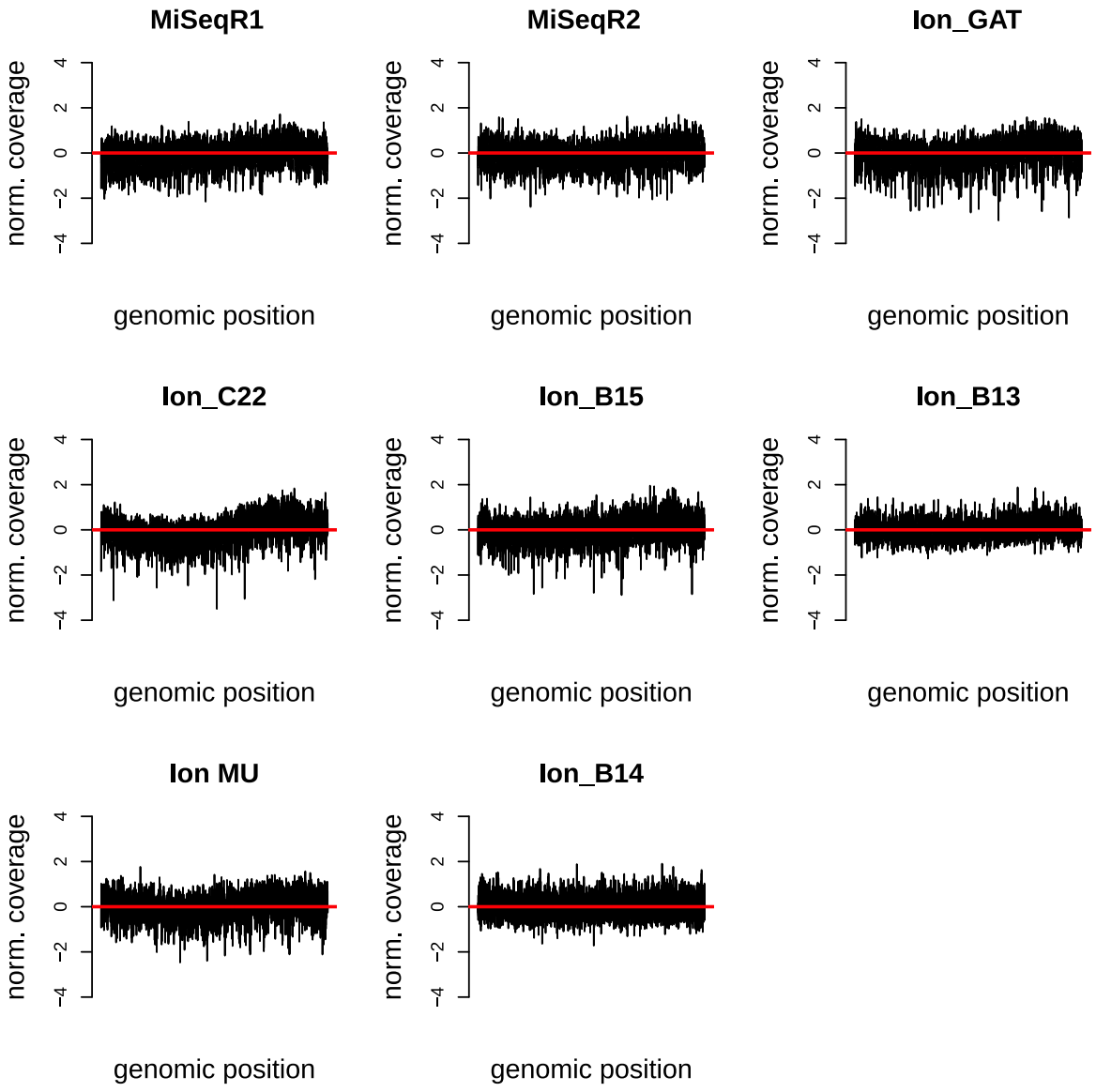

Figure 5.6: The averaged coverage of reads accross the reference genome of all 8 data sets. The coverage was adjusted using a Z transformation such that the mean = 0 and standard deviation = 1.

Chapter 6

DeFenSe: Detection of Falsely Aligned Sequences

6.1 Introduction

In the previous chapters we discussed two important topics when designing a Next Generation Sequencing (NGS) study. In chapter 3, we showed how to process sequencing data if a reference genome is available. In addition, we discussed methods to speed up the process of aligning the reads to a reference sequence in chapter 4. We have discussed how to assess the limitations and errors of NGS technologies (chapter 5). This is important to choose the appropriate technology for every NGS related experiment.

Another important aspect of every NGS related experiment is how to detect reads that are mapped but did not origin from the intended genome, e.g. an E. coli read is mapped when sequencing human, or belong to regions that are not present in the reference genome, e.g. due to genomic rearrangements. This signal has the potential for misinterpretation and classification of the data. In this chapter, we suggest one possible method to detect and remove the number of falsely aligned reads. With the possibility to align reads also to highly polymorphic regions the risk increases that reads are mapped although they originate from a different genome than intended to be sequenced. As one example, we may sequence an *Arabidopsis thal.* genome but observe also reads originating from an *Escherichia coli* genome. We refer to such reads as contaminating reads. In addition, medium or large insertion events, e.g. retro elements, lead to genomic regions that are sequenced but not present in the reference genome citepLee2012.

Aligning those reads will lead to a wrong signal in the data analysis, which may result in misinterpretation of the data. To solve this problem, we start with the assumption that all reads that are either contaminating reads or are due to genomic insertions have a higher number of differences compared to reads that origin from a genomic region that is present in the reference sequence. Differences are defined as the number of mismatches, insertions and deletions per read. We refer to wrongly mapped reads as reads that are mapped but are either contaminating reads or based on genomic insertion events or were not positioned on the location they originated from. Correctly mapped reads are reads that are mapped to the genomic region where they are sequenced from.

There are four different filtering approaches for detecting wrongly mapped reads in the field of NGS analysis. An identity threshold is used to reject reads that have a higher number of differences than specified by the user. Identity is computed for each alignment based on the number of matches divided by the alignment length. Alternatively a mapping quality (Li *et al.*, 2008) threshold is applied to reject those reads where the number of mismatches between the best scoring alignment and all observed alignments weighted by the quality values per mismatch position is lower than specified by the user. Note that both filter methods relies on ad hoc decisions or values for a given experiment. The two other methods are published and specifically address the reduction of contaminating reads. ContEst (Cibulskis *et al.*, 2011) uses prior knowledge of the genotype of the sequenced organism to filter the inferred variant sites. The other method introduced by Schmieder and Edwards (2011) relies on prior knowledge of the organism and genomic sequence where the contaminating reads origin from. It filters the reads based on a mapping to the reference genome of the organism that caused the contamination.

To reduce wrongly mapped reads the objective is to reject those reads that map with a higher number of differences than expected. This will be reflected in the alignment score for each read. The range of the alignment scores for correctly mapped reads is hard or nearly impossible to compute and is therefore unknown. This is because the optimal alignment score threshold is depending on the sequencing error, the genetic difference between the sequenced genome and the reference genome, the read length, the alignment method and other factors that we can not estimate and may vary between sequencing runs. Nevertheless, it is possible to compute the alignment score distribution of randomly mapped reads. Given this score distribution we can test if an alignment

score observed after mapping is significantly higher/better than the alignment scores of randomly mapped reads.

Here we present *DeFenSe* (Detection of Falsely aligned Sequences) an approach that computes the score distribution per data set of randomly mapped reads and uses the gathered information to reject wrongly mapped reads. *DeFenSe* works independently of adhoc decisions and does not require prior knowledge.

6.2 Methods

DeFenSe is divided into two parts. First, it computes the alignment score distribution of randomly mapped reads. Second, it computes the alignment score for the mapped reads and tests if the alignment score is significantly higher, than expected given the random score distribution.

6.2.1 Determination of the alignment score threshold

To determine the alignment score distribution *DeFenSe* randomly selects n reads ($n \approx 100,000$). Subsequently a relaxed candidate search strategy is applied. The candidate search is based on seed words of length k, by default $k = 12$. First, on every second position of the reference genome a seed word is extracted and its corresponding genomic positions is stored in a hash table. Second, every read is splitted in overlapping seed words. For each seed word all corresponding genomic positions are extracted from the hash table. Based on the relative position of the seed word in the read and the position in the genome a region of length $l = ||read|| + c$ is extracted, where $c > 0$ determines the number of allowed consecutive insertions or deletions and $||read||$ is equal to the read length. For every region an alignment score is computed based on a scoring function s. The resulting alignment score is stored. This is done $n \times m$ times, where m are all locations of all observed seed words per read. *DeFenSe* uses MASon (chapter 4). This allows *DeFenSe* to fully utilize CPUs and if available a graphic card to speed up the alignment score computation. Finally, *DeFenSe* reports the threshold, which is the 95% quantile of the obtained alignment score distribution. This value represents the lower alignment scoring border of reads that are not mapped by chance.

To test each read based on the before computed alignment score value, *DeFenSe* recomputes the alignment score given the reported CIGAR string and the used reference genome (Li *et al.*, 2009). A CIGAR string stores the necessary information to reconstruct the alignment reported by the mapping method. For each mapped read *DeFenSe* first reconstructs the alignment using the available information in the SAM entry. Next it computes the corresponding alignment score based on the scoring schema s. If the obtained alignment score is lower than the 95% quantile of the random mapping alignment score distribution, the read is rejected.

6.2.2 Evaluation

To evaluate the performance of *DeFenSe* we used real data from the Sequence Read Archive (SRA) of NCBI and introduced a contamination using sequencing reads from *Escherichia coli*. Table 6.1 summarizes the data. Every data set (D_1, D_2, D_3, C_1) was mapped with two different programs, SSaha2 (Ning *et al.*, 2001) and SHRiMP2 (David *et al.*, 2011) using their default or recommended parameter settings. In the evaluation, we focused on the number of uniquely mapped reads using SHRiMP2 both with local (SHRiMP2_lo) and global (SHRiMP2_gl) alignment options, and SSaha2, which uses a local alignment approach. After mapping, *DeFenSe* was used to screen the aligned reads. To compare the results from *DeFenSe* to other approaches, we screened the mapped data sets using an identity threshold. We computed identity based on the number of matches divided by the alignment length. Here, we choose a 95% ($ident_{95}$) and a 90% ($ident_{90}$) identity threshold to filter the mapped reads. Based on the computed mapping quality we screened the data set requiring a mapping quality > 20 ($mapQV_{20}$) (Li *et al.*, 2008). The mapping quality is computed by the majority of mapping programs and gives information on the confidence of the mapped read. Most of the time people recommend to filter for this value to obtain a reliable mapping results.

	Dataset	Organism	Experiment	number of reads	read length (bp).
D_1	SRR013327	*Arabidopsis thal.*	Genome-Seq	61,105,690	40
	SRR013328	*Arabidopsis thal.*	Genome-Seq	57,810,956	40
D_2	SRR333517	*Drosophila mel.*	Genome-Seq	57,316,770	40
D_3	SRR063698	*Drosophila mel.*	Genome-Seq	56,154,204	95
C_1	ERR023717	*Mus musculus*	ChIP-Seq	37,746,682	36
	SRR364816	*Escherichia coli*	Genome-Seq	30,656,098	101

Table 6.1: Data sets used for evaluation obtained from NCBI SRA. All non *Escherichia coli* data sets were contaminated with 30.7 million *Escherichia coli* reads of the corresponding read length.

6.3 Results

6.3.1 Genome-Seq: *Arabidopsis thaliana* short reads

D_1 consists of 118.92 million reads with length 40bp from *Arabidopsis thaliana* and 30.66 million reads from an *Escherichia coli* Genome-Sequencing experiment. This corresponds to a 20.50% contamination. The mapping results are summarized in Table 6.2.

SHRiMP2 using a global alignment (SHRiMP2_gl) uniquely mapped 54.48% of *Arabidopsis thal.* reads. In addition, it aligned 0.20% (62,558 reads) of the *Escherichia coli* reads. SHRiMP2 using a local alignment (SHRiMP2_lo) approach mapped 56.48% of the *Arabidopsis thal.* reads and 1.50% (459,189 reads) of the *Escherichia coli* reads. SSaha2 mapped the most *Arabidopsis thal.* reads (66.62%). However, it also mapped 49.84% of the *Escherichia coli* reads.

Applying $ident_{95}$ we could reduce the fraction of mapped *Escherichia coli* reads in both SHRiMP2 approaches (global: 1.04×10^{-3}% and local: 0.33%). However, at the same time the percentages of *Arabidopsis thal.* reads were reduced to 35.23% for SHRiMP2_gl and to 39.90% for SHRiMP2_lo. When filtering using $ident_{95}$ for the data set mapped with SSaha2 reduced the fraction of aligned *Escherichia coli* reads from 49.84% to 47.07%. For the *Arabidopsis thal.* reads we measured a reduction from 66.62% to 64.44%.

The mapping quality per mapped read is only available for SSaha2. When applying $mapQV_{20}$ the fraction of mapped *Escherichia coli* reads dropped to 0.03%. In addition, the percentage of mapped *Arabidopsis thal.* reads were reduced from 66.62% to 36.66%.

Subsequently, we filtered the results with *DeFenSe*. Figure 6.1 shows the resulting alignment score distribution (green) for data set D_1. The alignment score distribution and the resulting score threshold given a 95% quantile for the local alignment (threshold = 95) was computed in 3 minutes. We obtained a similar runtime for computing the score threshold for the global alignment threshold (threshold = 92). We next determined the alignment score distribution for the reads mapped by SHRiMP2_lo and SSaha2 (Figure 6.1 A and B respectively). Figure 6.2 shows the score distribution plus the scoring thresholds for each mapping method.

Using the global alignment threshold computed by *DeFenSe* for SHRiMP2_gl we measured a reduction of *Escherichia coli* reads from 0.20% to 0.19%. The *Arabidopsis thal.* reads were reduced from 54.48% to 54.28%. For the results of SHRiMP2_lo we observed a reduction from 1.5% to 0.66% of mapped *Escherichia coli* reads. For *Arabidopsis thal.*, we measured a reduction from 56.48% to 55.64% of mapped reads. When filtering the mapped reads from SSaha2 we observe a reduction of *Escherichia coli* reads from 49.84% to 0.20%. For the *Arabidopsis thal.* reads 50.73% were preserved.

	SHRiMP2_gl $_{global\,aln.}$(%)		SHRiMP2_lo $_{local\,aln.}$(%)		SSaha2 $_{local\,aln.}$(%)	
Organism	*Drosophila mel.*	*Escherichia coli*	*Drosophila mel.*	*Escherichia coli*	*Drosophila mel.*	*Escherichia coli*
uniq. mapped	54.48	0.20	56.48	1.5	66.62	49.84
$ident_{95}$	35.23	$\mathbf{1.04 \times 10^{-3}}$	39.90	**0.33**	55.83	47.07
$ident_{90}$	44.16	0.02	49.44	0.86	**64.44**	49.64
$mapQV_{20}$	NA	NA	NA	NA	36.66	**0.03**
DEFENSE	**54.28**	0.19	**55.64**	0.66	50.73	0.20

Table 6.2: Percentage of uniquely mapped reads for D_1 for all three mappers. The Table shows the result of filtering the mapped reads using different identity and mapping quality thresholds. *DEFENSE* computed the filtering threshold based on the raw reads.

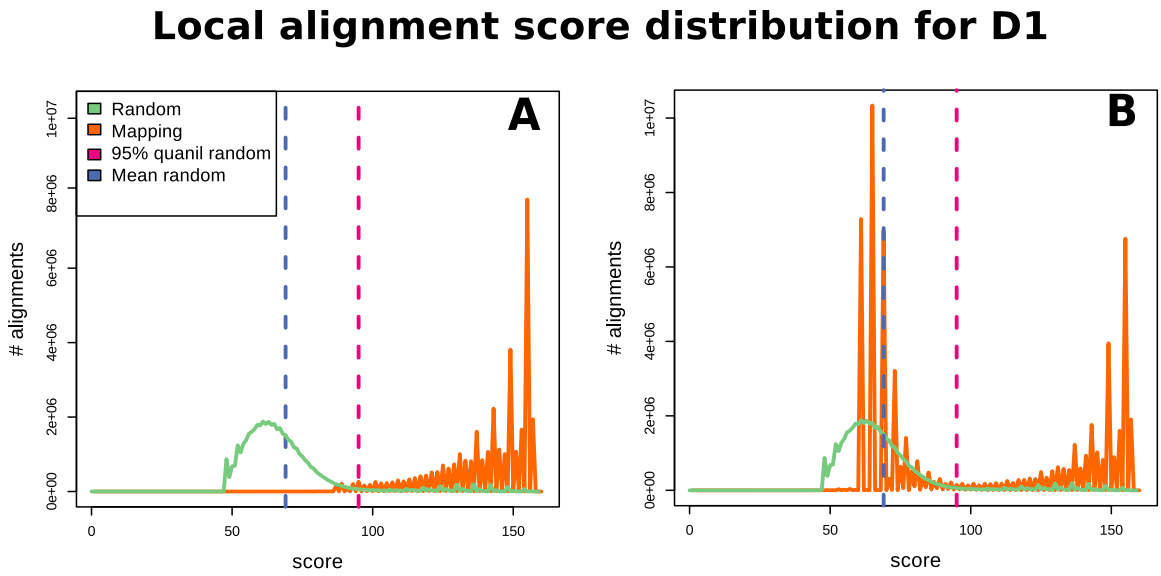

Figure 6.1: Local alignment score histograms of D_1. The 95% quantile threshold (red) of the randomly mapping score distribution (green) was used to filter the mapped (orange) reads of SHRiMP2_lo (A) and SSaha2 (B)

6.3.2 Genome-Seq: *Drosophila mel.* short reads

D_2 consists of 57.32 million reads (40bp long) from *Drosophila melanogaster* and 30.66 million reads from *Escherichia coli*. This corresponds to a 34.84% contamination. D_2 differs from D_1 since *Drosophila mel.* has a higher genetic variability than *Arabidopsis thal.* (Nordborg *et al.*, 2005; Mackay *et al.*, 2012). In addition, the contamination rate is increased resulting in a more challenging data set for filtering. The mapping results of the individual methods are shown in Table 6.3.

SHRiMP2_gl aligned 85.28% of the *Drosophila mel.* reads and 0.37% of the *Escherichia coli* reads. For mapping with SHRiMP2_lo we got 85.88% *Drosophila mel.* reads and 12.65% of the *Escherichia coli* reads. The mapping of SSaha2 again resulted in the highest percentage of uniquely mapped reads. For *Drosophila mel.* we measured 85.91% uniquely mapped reads but at the same time 56.64% of the *Escherichia coli* reads were uniquely mapped.

The results for applying $ident_{95}$ and $ident_{90}$ are shown in Table 6.3. When applying $ident_{95}$ on the mapped data set from SHRiMP2_gl 84.90% of the *Drosophila mel.* reads remained. For *Escherichia coli*, 0.32% (97,588 reads) of the reads remained. For

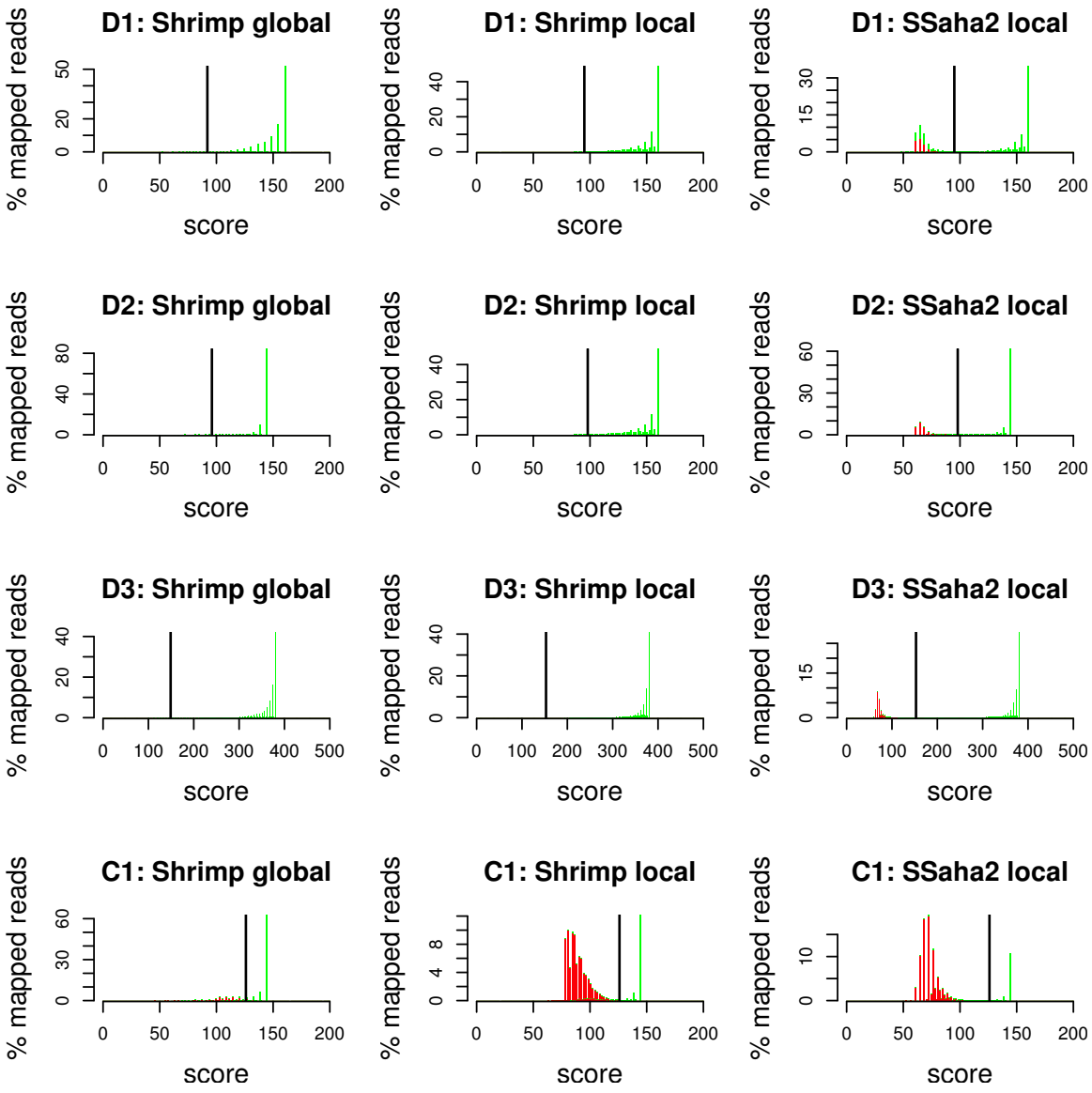

Figure 6.2: Score histograms of the 4 selected data sets. Marked in red are those reads that origin from the *Escherichia coli* data set. In green, those reads from the original data set. The black line marks the score cutoff.

SHRiMP2_lo we measured 81.92% of uniquely mapped *Drosophila mel.* reads. The $ident_{95}$ for SHRiMP2_lo reduced the mapped *Escherichia coli* reads from 12.65% to 8.86%. Mapping with SSaha2 and applying $ident_{95}$ resulted in 83.46% of mapped *Drosophila mel.* reads and 53.35% of *Escherichia coli* reads.

When filtering according to $mapQV_{95}$ 74.91% of uniquely mapped *Drosophila mel.*

	SHRiMP2_gl $_{global\,aln.}$(%)		SHRiMP2_lo $_{local\,aln.}$(%)		SSaha2 $_{local\,aln.}$(%)	
Organism	*Drosophila mel.*	*Escherichia coli*	*Drosophila mel.*	*Escherichia coli*	*Drosophila mel.*	*Escherichia coli*
uniq. mapped	85.28	0.37	85.88	12.65	85.91	56.64
$ident_{95}$	79.40	$\mathbf{2.09 \times 10^{-4}}$	81.92	8.86	83.46	53.35
$ident_{90}$	83.44	3.28×10^{-3}	85.54	8.86	**85.77**	56.46
$mapQV_{20}$	NA	NA	NA	NA	74.91	$\mathbf{9.09 \times 10^{-3}}$
DEFENSE	**84.90**	0.32	**85.15**	**0.47**	84.38	0.07

Table 6.3: Percentage of uniquely mapped reads for D_2 for all three mappers. The Table shows the filtering of the mapped reads using different identity and mapping quality thresholds. *DEFENSE* computed the filtering threshold based on the raw reads.

reads were preserved for the mapping of SSaha2. Only 0.01% (2,787) of *Escherichia coli* reads remained.

DeFenSe computed the score distribution for randomly mapped reads in less than 5 minutes based on 67 million alignment scores. The score distribution for all three mapping results plus the filtering decision are shown in Figure 6.2. For SHRiMP2_gl 84.90% of the *Drosophila mel.* reads and 0.32% of the *Escherichia coli* reads remained. Using SHRiMP2_lo 85.15% of the *Drosophila mel.* reads were preserved while the *Escherichia coli* reads were reduced from 12.65% to 0.47%. A similar picture is observed when filtering the result of the SSaha2 mapping. Here, 84.38% of the *Drosophila mel.* reads were preserved while the *Escherichia coli* reads were reduced from 56.64% to 0.07%. *DeFenSe* maintained 98.23% of the mapped *Drosophila mel.* reads while rejecting 99.88% of the *Escherichia coli* reads.

6.3.3 Genome-Seq: *Drosophila mel.* long reads

Data set D_3 consists of 56.15 million reads with 95bp from *Drosophila melanogaster*. The 30.66 million reads from *Escherichia coli* were sampled with the respective read length. This results in 35.31% contamination. D_3 differs from D_2 by an increase of

the read length by a factor of 2. The mapping results are shown in Table 6.4 for the programs SHRiMP2_gl (global alignment), SHRiMP2_lo (local alignment) and SSaha2 respectively.

SHRiMP2_gl mapped 79.02% of the *Drosophila mel.* reads. For the local alignment method we measured that 80.51% of the *Drosophila mel.* reads were uniquely mapped to the reference genome. Both instances of SHRiMP2 mapped nearly non (global: 4.63×10^{-3}% and local: 3.59×10^{-3}%) of the *Escherichia coli* reads. This is probably due to the larger read length of 95bp. SSaha2 mapped again the highest percentage (83.32%) of *Drosophila mel.* reads. However, a high percentage of *Escherichia coli* reads (55.87%) were also mapped.

When filtering with $ident_{95}$ we observed a reduction of mapped *Drosophila mel.* reads for the mapping of SHRiMP2_gl from 80.51% to 67.56%. For the local alignment method the percentage is decreased from 79.02% to 62.67% of uniquely mapped *Drosophila mel.* reads. Filtering the mapping of SSaha2 resulted in a reduction of *Drosophila mel.* reads from 83.32% to 72.16%. At the same time, the percentage of *Escherichia coli* reads were reduced from 55.87% to 50.17%.

Filtering the mapping result of SSaha2 according to $mapQV_{20}$ resulted in a reduction of *Drosophila mel.* reads from 83.32% to 74.88%. The percentage of *Escherichia coli* reads were reduced from 55.87% to 50.17%.

DeFenSe required 12 minutes to compute the score distribution based on 243 million alignment scores. The score distribution for the mapped reads plus the scoring threshold are shown in Figure 6.2. For SHRiMP2_gl *DeFenSe* reduced the percentage of mapped *Drosophila mel.* reads from 79.02% to 78.83%. For SHRiMP2_lo, *DeFenSe* preserved all *Drosophila mel.* reads. When filtering the results of SSaha2 *DeFenSe* reduced the percentage of mapped *Drosophila mel.* reads from 83.32% to 81.35% while remaining 0.02% of mapped *Escherichia coli* reads.

6.3.4 ChIP-Seq: *Mus musculus*

Data set C_1 consists of 37.75 million reads of 36bp from a ChIP-Seq experiment of *Mus musculus*. The *Escherichia coli* reads were sampled with the respective read length. This results in a contamination of 44.82%. The mapping results are shown in the Table

	SHRiMP2_gl $_{global\,aln.}$(%)		SHRiMP2_lo $_{local\,aln.}$(%)		SSaha2 $_{local\,aln.}$(%)	
Organism	*Drosophila mel.*	*Escherichia coli*	*Drosophila mel.*	*Escherichia coli*	*Drosophila mel.*	*Escherichia coli*
uniq. mapped	79.02	4.63×10^{-3}	80.51	3.59×10^{-3}	83.32	55.87
$ident_{95}$	62.67	0.00	67.56	$\mathbf{3.26 \times 10^{-5}}$	72.16	50.17
$ident_{90}$	72.66	$\mathbf{1.60 \times 10^{-4}}$	76.64	7.50×10^{-5}	80.80	55.13
$mapQV_{20}$	NA	NA	NA	NA	74.88	$\mathbf{3.89 \times 10^{-3}}$
DEFENSE	**78.83**	4.63×10^{-3}	**80.51**	3.59×10^{-3}	**81.35**	0.02

Table 6.4: Percentage of uniquely mapped reads for D_3 for all three used mappers. The Table shows the filtering of the mapped reads using different identity and mapping quality thresholds. *DEFENSE* computed the filtering threshold based on the raw reads.

6.5 for the programs SHRiMP2_gl (global alignment), SHRiMP2_lo (local alignment) and for SSaha2.

When mapping the reads with SHRiMP2_gl 70.31% of the *Mus musculus* and 2.22% of the *Escherichia coli* reads were uniquely mapped. SHRiMP2_lo resulted in an increased percentage of mapped *Mus musculus* (76.98%) and *Escherichia coli* (44.26%) reads compared to the global alignment version. SSaha2 mapped 74.89% of *Mus musculus* reads while again mapping the highest percentage of mapped *Escherichia coli* reads of 51.95%.

Filtering the results of SHRiMP2_gl using $ident_{95}$ resulted in a reduction from 70.31% to 61.24% of *Mus musculus* reads. The *Escherichia coli* reads were reduced from 2.22% to 1.38×10^{-3}%. After filtering ($ident_{95}$), the mapping result of SHRiMP2_lo 66.29% of the *Mus musculus* reads were preserved while 15.04% of the *Escherichia coli* reads remained. When applying $ident_{95}$ on the mapping results for SSaha2 70.91% of the mapped *Mus musculus* and 48.31% of the mapped *Escherichia coli* reads remained

When applying $mapQV_{20}$ on the mapping result from SSaha2 only 50.28% of the *Mus musculus* remained. At the same time, only 3.89×10^{-3}% of the *Escherichia coli* reads remained.

DeFenSe required 76 minutes to compute the score threshold based on 736 million.

	SHRiMP2_gl *global aln.* (%)		SHRiMP2_lo *local aln.* (%)		SSaha2 *local aln.* (%)	
Organism	*Drosophila mel.*	*Escherichia coli*	*Drosophila mel.*	*Escherichia coli*	*Drosophila mel.*	*Escherichia coli*
uniq. mapped	70.31	2.22	76.98	44.26	74.89	51.95
$ident_{95}$	61.24	$\mathbf{1.38 \times 10^{-3}}$	66.29	15.04	70.91	48.31
$ident_{90}$	**65.29**	0.02	**72.85**	30.96	**74.40**	51.58
$mapQV_{20}$	NA	NA	NA	NA	50.28	$\mathbf{1.67 \times 10^{-3}}$
DEFENSE	63.81	6.55×10^{-3}	64.32	**0.01**	63.92	1.61×10^{-3}

Table 6.5: Percentage of uniquely mapped reads for C_1 for all three used mappers. The Table shows the filtering of the mapped reads using different identity and mapping quality thresholds. *DEFENSE* computed the filtering threshold based on the raw reads.

alignment scores. The score distribution plus the scoring thresholds are shown in Figure 6.2. When filtering the result from SHRiMP2_gl we measured a reduction from 70.31% to 63.81% for the mapped *Mus musculus* while nearly all *Escherichia coli* reads were rejected (6.55×10^{-3}% remaining). 64.32% of the *Mus musculus* reads remained after filtering the outcome of SHRiMP2_lo. The percentage of *Escherichia coli* reads were reduced from 44.26% to 0.01%. Filtering the results of SSaha2 resulted in a reduction from 74.89% to 63.92% of *Mus musculus* reads. At the same time, nearly all *Escherichia coli* reads (1.61×10^{-3}% remaining) were rejected.

6.4 Discussion

One of the first steps when analyzing NGS data is to align the reads to a reference genome, if existing. Various mapping methods have been published aiming either at a short runtime or on mapping reads in highly polymorphic regions. To align reads into high polymorphic regions, a mapping method has to allow for a high number of differences between the read and the reference sequence. Allowing for a high number of differences increases the risk of a read being mapped although it original location is not present on the reference sequence. This is the case, when a read origins either

from a different species or from a genomic location that is not present due to genomic rearrangements. In such a scenario, the mapping of contaminating reads will lead, for example, to a wrong identification of SNPs, depending on the analysis and the experiment, which results in a wrong or not reliable interpretation of the data.

The method presented here detects such contaminating reads. In general, *DeFenSe* is based on the assumption that contaminating reads or generally speaking reads that are mapped to a wrong location in the reference genome, show a higher number of differences between the read and the reference compared to reads that are mapped to the correct position. Based on this assumption it is theoretically possible to compute the score distribution of correctly mapped reads. However, this is in practice nearly impossible since one has to correctly estimate several parameters including sequencing, mapping and preparation errors. Therefore, we compute the score distribution of randomly mapped reads. This represents those reads that are mapped by chance. Given this score distribution we use the 95% quantile to reject those reads that are mapped by chance (having a lower score) given the mapped data. Note that this works independent of the mapping method, sequencing error rate, contamination rate and differences between the sequenced and the reference genome.

We assessed the performance of *DeFenSe* by comparing it to the filtering based on the identity and the mapping quality of the mapped reads. In contrast to *DeFenSe* the identity is depending on the overall error rate and the distance between the sequenced genome and the reference genome. This could be shown when comparing data sets D_1 and D_2, which have both the same read lengths and were sequenced with the same technology but differ in the sequenced organisms. In case of *Arabidopsis thal.*, using the $ident_{95}$ resulted in aligning only 0.33% of the *Escherichia coli* reads, whereas in the case of *Drosophila mel.* 8.86% of the *Escherichia coli* reads remained present after filtering. Note that the same filtering process gives two completely different results in terms of avoiding contaminating reads. In contrast to *DeFenSe*, where only 0.66% (D_1) and 0.47% (D_2) of the *Escherichia coli* reads remained after filtering the mapped data of SHRiMP2_lo. Furthermore, $ident_{95}$ completely failed on filtering results obtained from SSaha2. SSaha2 is a local alignment based mapping method. This is because SSaha2 seems to reduce the alignment length to such a level, where only very few differences are present. The approach fails here because the identity is computed based on the alignment length and the number of differences within the alignments. In contrast,

DeFenSe also uses the information of the length of the alignment in comparison to the original read length by looking at the alignment score.

The mapping quality (Li *et al.*, 2008) is computed based on the number of mismatches and their quality values given the best scoring alignment in comparison to the quality values of mismatch positions of all observed alignments per read. The assumption is that a mismatch is most of the time related to a low quality value of a particular base. As shown filtering according to the mapping quality works. However, when it comes to polymorphic regions we have the problem that most of the mismatches have a high quality value. Therefore, reads assigned to such regions will probably get a low mapping quality score and might subsequently be filtered out. In all of the data sets, we observed that *DeFenSe* preserved a higher percentage of mapped reads from the organism that was sequenced. For example, when filtering D_2 using the mapping quality we observe a reduction from 85.91% (SSaha2) of mapped *Drosophila mel.* reads down to 74.91%. When filtering with *DeFenSe* additional 10% (84.38%) of the *Drosophila mel.* reads were preserved. For both instances of SHRiMP2 we did not obtain a mapping quality for the alignments.

While filtering using *DeFenSe* obviously improved the results for mapping with SSaha2 the results using both SHRiMP2 versions are not striking. Still, in case of the local alignment version we could demonstrate that *DeFenSe* always reduces the percentage of bacterial reads while maintain the percentage of reads originated by the used reference sequence. For the global alignment version of SHRiMP2 the identity threshold identifies more bacterial reads compared to *DeFenSe*. However, this comes with the disadvantage of rejecting up to 20% (D_1) of those reads that originated from the same organism as the reference sequence compared to the result of *DeFenSe*.

The current version of *DeFenSe* can not cope with different raw read lengths. This is because the alignment score is influenced by different read lengths. One expects that shorter reads lead to a reduced alignment score compared to longer reads. Therefore, all shorter reads will automatically have the risk to be rejected. This issue will be solved in the next release of *DeFenSe* and enable the usage for other technologies besides from Illumina. Another requirement of *DeFenSe* is that the sampled distribution has to be more or less normally distributed. If this is not the case, for example, if the relaxed mapping strategy always identifies only the best scoring position, then the method does not work correctly. The next version of *DeFenSe* will report the probability of malfunctioning on

the sample process.

To conclude, *DeFenSe* works based on the underlying raw reads and is independent of ad hoc decisions and prior knowledge. This makes it interesting for non model organisms where we typically have minor or no knowledge available. For model organisms we suggest a combination of tools that are already available (e.g. ContEst (Cibulskis *et al.*, 2011)) and *DeFenSe*. This allows to apply prior knowledge in combination with a method to reduce wrongly mapped reads and improve the reliability of NGS results.

Chapter 7

Summary

In this thesis we have addressed four relevant topics for the analysis of Next Generation Sequence (NGS) analysis.

Firstly, we have addressed the problem of mapping NGS reads to a reference genome (chapter 3). Here we focused on maximizing the number of accurately placed reads, independently of read length and distance from the reference genome. This requires a search strategy to align reads without a cut off for the number of mismatches, insertions and deletions. This is becoming more important because new technologies produce longer reads with an increased absolute number of sequence differences and many of newly sequenced organisms have a considerably large evolutionary distance to reference genomes. We demonstrated the advantages of this method based on real and simulated data.

Secondly, we optimized a local pairwise alignment algorithm for the use in the score of NGS analysis (chapter 4) and showed its advantage in terms of runtime over the standard implementation. In addition, we implemented the algorithm on various hardware platforms and compared currently available high performance technologies like vector instruction sets on the CPU and optimizations on the GPU. The result of this was an optimized library implementation for pairwise alignments, which was used in the chapters 3 and 6.

Third, we have presented methods to characterize technical biases of current sequencing technologies, exemplified on two benchtop technologies: Ion Torrent and MiSeq (chapter 5). We discovered a bias in the nucleotide frequencies of sequencing errors as well as a spatial clustering of reads with an increased error rate related to a location on

the sample carrier in the sequencing machine. These and other characterizations (e.g. different error sources: mismatches vs. indels) presented in this thesis, revealed issues that should be taken into account when choosing the appropriate sequencing technology for an experiment.

Due to the sensitivity of NGS technology not only DNA from the targeted sample is sequenced, but also contaminating reads occur (e.g. *E. coli* in a human sample). These reads can represent serious problems for biological inference. Here we presented a method that detects and excludes contaminating reads (chapter 6). In contrast to previous approaches, this method automatically adapts to the underlying genetic diversity between the reference and the sequenced genome as well as the sequencing error rate. This makes it applicable to the analysis of model and non-model organisms. We demonstrated its utility with real data for different data from organisms and varying read lengths.

All methods and programs are open source (Artistic License (Perl) 1.0). For documentation and download see: http://www.cibiv.at/software/ngm

Program	Description
NextGenMap	A program to map NGS reads to a reference genome/ transcripts (chapter 3)
NextGenBench	Benchmark system for NGS mapper (chapter 3)
MASon (Million Alignments within Seconds)	C++ optimized alignment library package for NGS (chapter 4)
MoNTy (Measurements for NGS Technology)	Performs several measurements and plots to reveal the quality of a sequenced data set (chapter 5)
DeFenSe (Detection of Falsely Aligned Sequences)	Method to detect and reduce contamination in NGS data (chapter 6)

Bibliography

1000 Genomes Project Consortium (2010) A map of human genome variation from population-scale sequencing. *Nature*, **467**, 1061–1073.

Altschul, S. F., Gish, W., Miller, W., Myers, E. W. and Lipman, D. J. (1990) Basic local alignment search tool. *Journal of molecular biology*, **215**, 403–410.

Burrows, M. and Wheeler, D. J. (1994) A block-sorting lossless data compression algorithm. Tech. Rep. 124.

Cannings, C. and Edwards, A. W. (1968) Natural selection and the de finetti diagram. *Annals of human genetics*, **31**, 421–428.

Cartwright, R. A. (2005) DNA assembly with gaps (Dawg): simulating sequence evolution. *Bioinformatics*, **21**, iii31–iii38.

Chaisson, M. J., Brinza, D. and Pevzner, P. A. (2009) De novo fragment assembly with short mate-paired reads: Does the read length matter? *Genome Research*, **19**, 336–346.

Cibulskis, K., McKenna, A., Fennell, T., Banks, E., DePristo, M. and Getz, G. (2011) ContEst: estimating cross-contamination of human samples in next-generation sequencing data. *Bioinformatics*, **27**, 2601–2602.

Cock, P. J. A., Fields, C. J., Goto, N., Heuer, M. L. and Rice, P. M. (2010) The sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. *Nucleic Acids Research*, **38**, 1767–1771.

Dagum, L. and Menon, R. (1998) OpenMP: an industry standard API for shared-memory programming. *IEEE Computational Science and Engineering*, **5**, 46–55.

Darling, A. E., Mau, B. and Perna, N. T. (2010) progressiveMauve: Multiple genome alignment with gene gain, loss and rearrangement. *PLoS ONE*, **5**, e11147+.

David, M., Dzamba, M., Lister, D., Ilie, L. and Brudno, M. (2011) SHRiMP2: Sensitive yet practical short read mapping. *Bioinformatics*, **27**, 1011–1012.

Döring, A., Weese, D., Rausch, T. and Reinert, K. (2008) SeqAn an efficient, generic C++ library for sequence analysis. *BMC bioinformatics*, **9**, 11.

Farrar, M. (2007) Striped SmithâĂŞWaterman speeds database searches six times over other SIMD implementations. *Bioinformatics*, **23**, 156–161.

Flicek, P. and Birney, E. (2009) Sense from sequence reads: methods for alignment and assembly. *Nature Methods*, **6**, S6–S12.

Glenn, T. C. (2011) Field guide to next-generation DNA sequencers. *Molecular Ecology Resources*, **11**, 759–769.

Gusfield, D. (1997) *Algorithms on strings, trees, and sequences : computer science and computational biology*. Cambridge Univ. Press.

Hall, N. (2007) Advanced sequencing technologies and their wider impact in microbiology. *J Exp Biol*, **210**, 1518–1525.

Harismendy, O., Ng, P., Strausberg, R., Wang, X., Stockwell, T., Beeson, K., Schork, N., Murray, S., Topol, E., Levy, S. and Frazer, K. (2009) Evaluation of next generation sequencing platforms for population targeted sequencing studies. *Genome Biology*, **10**, R32+.

Holt, R. A. and Jones, S. J. (2008) The new paradigm of flow cell sequencing. *Genome research*, **18**, 839–846.

Homer, N., Merriman, B. and Nelson, S. F. (2009) BFAST: An alignment tool for large scale genome resequencing. *PLoS ONE*, **4**, e7767+.

Huse, S., Huber, J., Morrison, H., Sogin, M. and Welch, D. (2007) Accuracy and quality of massively parallel DNA pyrosequencing. *Genome Biology*, **8**, R143+.

Illumina Inc. (2011a) .

Illumina Inc. (2011b) E. coli sequencing on the miseq system and ion torrent pgm system.

Jukes, T. H. and Cantor, C. R. (1969) Evolution of protein molecules. *Manmmalian Protein Metabolism*, pages 21–132.

Kent, J. J. (2002) BLAT - the BLAST-like alignment tool. *Genome research*, **12**, 656–664.

Kirk, D. B. and Mei, W. (2010) *Programming Massively Parallel Processors: A Hands-on Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, First edn..

Langmead, B., Trapnell, C., Pop, M. and Salzberg, S. (2009) Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biology*, **10**, R25+.

Li, C. and Rabinovic, A. (2007) Adjusting batch effects in microarray expression data using empirical bayes methods. *Biostatistics*, **8**, 118–127.

Li, H. and Durbin, R. (2009) Fast and accurate short read alignment with BurrowsâĂŞWheeler transform. *Bioinformatics*, **25**, 1754–1760.

Li, H. and Durbin, R. (2010) Fast and accurate long-read alignment with BurrowsâĂŞWheeler transform. *Bioinformatics*, **26**, 589–595.

Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., Marth, G., Abecasis, G., Durbin, R. and 1000 Genome Project Data Processing Subgroup (2009) The sequence Alignment/Map format and SAMtools. *Bioinformatics (Oxford, England)*, **25**, 2078–2079.

Li, H. and Homer, N. (2010) A survey of sequence alignment algorithms for next-generation sequencing. *Briefings in Bioinformatics*, **11**, 473–483.

Li, H., Ruan, J. and Durbin, R. (2008) Mapping short DNA sequencing reads and calling variants using mapping quality scores. *Genome Research*, **18**, 1851–1858.

Life Technogies Corp. (2011a) .

Life Technogies Corp. (2011b) The ion pgm sequencer exhibits superior long-read accuracy.

Liu, Y., Maskell, D. L. and Schmidt, B. (2009) CUDASW++: optimizing Smith-Waterman sequence database searches for CUDA-enabled graphics processing units. *BMC research notes*, **2**, 73+.

Loman, N. J., Misra, R. V., Dallman, T. J., Constantinidou, C., Gharbia, S. E., Wain, J. and Pallen, M. J. (2012) Performance comparison of benchtop high-throughput sequencing platforms. *Nature biotechnology*, **30**, 434–439.

MacCallum, D. M., Castillo, L., Nather, K., Munro, C. A., Brown, A. J. P., Gow, N. A. R. and Odds, F. C. (2009) Property differences among the four major candida albicans strain clades. *Eukaryot Cell*.

Mackay, T. F. C., Richards, S., Stone, E. A., Barbadilla, A., Ayroles, J. F., Zhu, D., Casillas, S., Han, Y., Magwire, M. M., Cridland, J. M., Richardson, M. F., Anholt, R. R. H., Barron, M., Bess, C., Blankenburg, K. P., Carbone, M. A., Castellano, D., Chaboub, L., Duncan, L., Harris, Z., Javaid, M., Jayaseelan, J. C., Jhangiani, S. N., Jordan, K. W., Lara, F., Lawrence, F., Lee, S. L., Librado, P., Linheiro, R. S., Lyman, R. F., Mackey, A. J., Munidasa, M., Muzny, D. M., Nazareth, L., Newsham, I., Perales, L., Pu, L.-L., Qu, C., Ramia, M., Reid, J. G., Rollmann, S. M., Rozas, J., Saada, N., Turlapati, L., Worley, K. C., Wu, Y.-Q., Yamamoto, A., Zhu, Y., Bergman, C. M., Thornton, K. R., Mittelman, D. and Gibbs, R. A. (2012) The drosophila melanogaster genetic reference panel. *Nature*, **482**, 173–178.

MacLean, D., Jones, J. D. G. and Studholme, D. J. (2009) Application of 'next-generation' sequencing technologies to microbial genetics. *Nature Reviews Microbiology*, **7**, 287–296.

Mardis, E. R. (2008) Next-Generation DNA sequencing methods. *Annual Review of Genomics and Human Genetics*, **9**, 387–402.

Margulies, M., Egholm, M., Altman, W. E., Attiya, S., Bader, J. S., Bemben, L. A., Berka, J., Braverman, M. S., Chen, Y.-J., Chen, Z., Dewell, S. B., Du, L., Fierro, J. M., Gomes, X. V., Godwin, B. C., He, W., Helgesen, S., Ho, C. H., Irzyk, G. P., Jando, S. C., Alenquer, M. L. I., Jarvie, T. P., Jirage, K. B., Kim, J.-B., Knight, J. R., Lanza, J. R., Leamon, J. H., Lefkowitz, S. M., Lei, M., Li, J., Lohman, K. L., Lu, H., Makhijani, V. B., McDade, K. E., McKenna, M. P., Myers, E. W., Nickerson, E., Nobile, J. R., Plant, R., Puc, B. P., Ronan, M. T., Roth, G. T., Sarkis, G. J., Simons,

J. F., Simpson, J. W., Srinivasan, M., Tartaro, K. R., Tomasz, A., Vogt, K. A., Volkmer, G. A., Wang, S. H., Wang, Y., Weiner, M. P., Yu, P., Begley, R. F. and Rothberg, J. M. (2005) Genome sequencing in microfabricated high-density picolitre reactors. *Nature*, **437**, 376–380.

Maxam, A. M. and Gilbert, W. (1977) A new method for sequencing DNA. *Proceedings of the National Academy of Sciences of the United States of America*, **74**, 560–564.

Medini, D., Serruto, D., Parkhill, J., Relman, D. A., Donati, C., Moxon, R., Falkow, S. and Rappuoli, R. (2008) Microbiology in the post-genomic era. *Nature Reviews Microbiology*, **6**, 419–430.

Metzker, M. L. (2010) Sequencing technologies - the next generation. *Nature reviews. Genetics*, **11**, 31–46.

Narzisi, G. and Mishra, B. (2011) Comparing de novo genome assembly: The long and short of it. *PLoS ONE*, **6**, e19175+.

Needleman, S. and Wunsch, C. (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, **48**, 443–453.

Niemenmaa, M., Kallio, A., Schumacher, A., Klemelä, P., Korpelainen, E. and Heljanko, K. (2012) Hadoop-BAM: directly manipulating next generation sequencing data in the cloud. *Bioinformatics*, **28**, 876–877.

Ning, Z., Cox, A. J. and Mullikin, J. C. (2001) SSAHA: a fast search method for large DNA databases. *Genome research*, **11**, 1725–1729.

Nordborg, M., Hu, T. T., Ishino, Y., Jhaveri, J., Toomajian, C., Zheng, H., Bakker, E., Calabrese, P., Gladstone, J., Goyal, R., Jakobsson, M., Kim, S., Morozov, Y., Padhukasahasram, B., Plagnol, V., Rosenberg, N. A., Shah, C., Wall, J. D., Wang, J., Zhao, K., Kalbfleisch, T., Schulz, V., Kreitman, M. and Bergelson, J. (2005) The pattern of polymorphism in arabidopsis thaliana. *PLoS Biol*, **3**, e196+.

Nvidia (2009) *NVIDIA CUDA C Programming Best Practices Guide*. Nvidia, First edn..

Ossowski, S., Schneeberger, K., Clark, R. M., Lanz, C., Warthmann, N. and Weigel, D. (2008) Sequencing of natural strains of arabidopsis thaliana with short reads. *Genome Research*, **18**, 2024–2033.

Pevzner, P. A., Tang, H. and Waterman, M. S. (2001) An eulerian path approach to DNA fragment assembly. *Proceedings of the National Academy of Sciences*, **98**, 9748–9753.

Ragazzini, J. R. and Zadeh, L. A. (1952) The analysis of sampled-data systems. *American Institute of Electrical Engineers*, **1**, 225–234.

Raman, S. K., Pentkovski, V. and Keshava, J. (2000) Implementing streaming SIMD extensions on the pentium III processor. *IEEE Micro*, **20**, 47–57.

Rothberg, J. M., Hinz, W., Rearick, T. M., Schultz, J., Mileski, W., Davey, M., Leamon, J. H., Johnson, K., Milgrew, M. J., Edwards, M., Hoon, J., Simons, J. F., Marran, D., Myers, J. W., Davidson, J. F., Branting, A., Nobile, J. R., Puc, B. P., Light, D., Clark, T. A., Huber, M., Branciforte, J. T., Stoner, I. B., Cawley, S. E., Lyons, M., Fu, Y., Homer, N., Sedova, M., Miao, X., Reed, B., Sabina, J., Feierstein, E., Schorn, M., Alanjary, M., Dimalanta, E., Dressman, D., Kasinskas, R., Sokolsky, T., Fidanza, J. A., Namsaraev, E., McKernan, K. J., Williams, A., Roth, G. T. and Bustillo, J. (2011) An integrated semiconductor device enabling non-optical genome sequencing. *Nature*, **475**, 348–352.

Rumble, S. M., Lacroute, P., Dalca, A. V., Fiume, M., Sidow, A. and Brudno, M. (2009) SHRiMP: Accurate mapping of short color-space reads. *PLoS Comput Biol*, **5**, e1000386+.

Sanger, F., Nicklen, S. and Coulson, A. R. (1977) DNA sequencing with chain-terminating inhibitors. *Proceedings of the National Academy of Sciences*, **74**, 5463–5467.

Schatz, M. C. (2009) CloudBurst: highly sensitive read mapping with MapReduce. *Bioinformatics*, **25**, 1363–1369.

Schbath, S., Martin, V., Zytnicki, M., Fayolle, J., Loux, V. and Gibrat, J.-F. (2012) Mapping reads on a genomic sequence: An algorithmic overwiew and a practical comparative analysis. *Journal of Computational Biology*, pages 120416083012005+.

Schmieder, R. and Edwards, R. (2011) Fast identification and removal of sequence contamination from genomic and metagenomic datasets. *PloS one*, **6**.

Shendure, J. and Ji, H. (2008) Next-generation DNA sequencing. *Nature Biotechnology*, **26**, 1135–1145.

Smith, T. F. and Waterman, M. S. (1981) Identification of common molecular subsequences. *Journal of Molecular Biology*, **147**, 195–197.

Stone, J. E., Gohara, D. and Shi, G. (2010) OpenCL: A parallel programming standard for heterogeneous computing systems. *Computing in Science & Engineering*, **12**, 66–73.

Strimmer, K. and von Haeseler, A. (1997) Likelihood-mapping: a simple method to visualize phylogenetic content of a sequence alignment. *Proceedings of the National Academy of Sciences of the United States of America*, **94**, 6815–6819.

Suzuki, S., Ono, N., Furusawa, C., Ying, B.-W. W. and Yomo, T. (2011) Comparison of sequence reads obtained from three next-generation sequencing platforms. *PloS one*, **6**, e19534+.

The International Human Genome Sequencing Consortium (2001) Initial sequencing and analysis of the human genome. *Nature*, **409**, 860–921.

Trapnell, C. and Salzberg, S. L. (2009) How to map billions of short reads onto genomes. *Nature Biotechnology*, **27**, 455–457.

Vouzis, P. D. and Sahinidis, N. V. (2011) GPU-BLAST: using graphics processors to accelerate protein sequence alignment. *Bioinformatics*, **27**, 182–188.

Zhang, W., Chen, J., Yang, Y., Tang, Y., Shang, J. and Shen, B. (2011) A practical comparison of de novo genome assembly software tools for Next-Generation sequencing technologies. *PLoS ONE*, **6**, e17915+.

Curriculum Vitae

Contact Information

Fritz Joachim Sedlazeck
Center for Integrative Bioinformatics Vienna (CIBIV)
Max F. Perutz Laboratories
Dr. Bohr Gasse 9
A-1030 Vienna, Austria
Phone: ++43 +1 / 4277-24023
Fax: ++43 +1 / 4277-24098
Email: fritz.sedlazeck(AT)univie.ac.at

Research Interests

- Sequence Analysis
- Structure prediction
- High performance computing

Education

- 1997 - 2002: ORG Antonkriegergasse, Austria
- 2004-2008: University of applied sciences Hagenberg, Austria
- Nov 2008 - : PhD student at the Center for Integrative Bioinformatics Vienna, Max F. Perutz Laboratories.

Degree

- DI (FH) (2008), Hagenberg, Austria Topic: Correlative Cryo-Fluorescence and cryo-Electron Microscopy.

Awards

- 2007 - 2008: Leonardo Da Vinci Scholarship

Professional Experience

- 09.2007-06.2008: Diploma Thesis at the EMBL Heidelberg, Frangakis Group, Topic: Implementation of Automatization Techniques for Correlative Cryo-Fluorescence and Cryo-Electron Microscopy

Publications

- P. Rescheneder, A. von Haeseler, and F.J. Sedlazeck (2011) MASon: Million Alignments In Seconds - A Platform Independent Pairwise Sequence Alignment Library for Next Generation Sequencing Data. Proceedings of the International Conference on Bioinformatics Models, Methods and Algorithms (BIOINFORMATICS 2012), 195-201, SciTePress, Setubal, Portugal. (DOI: 10.5220/0003775701950201)
- C. Veselye, S. Taubere, F.J. Sedlazeck, A. von Haeseler, and M.F. Jantsch (2012) Adenosine deaminases that act on RNA induce reproducible changes in abundance and sequence of embryonic miRNAs. Genome Res., 22, 1468-1476. (DOI: 10.1101/gr.133025.111, PMID: 22310477)
- H.Q. Dinh, M. Dubin, F.J. Sedlazeck, N. Lettner, O. Mittelsten Scheid, and A. von Haeseler (2012) Advanced Methylome Analysis after Bisulfite Deep Sequencing: an Example in Arabidopsis. PLoS ONE, 7, e41528. (DOI: 10.1371/journal.pone.0041528, PMCID: PMC3401099)