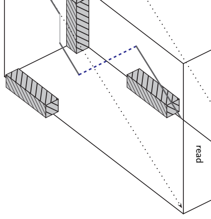


Genome Informatics for High-Throughput Sequencing Data Analysis - Methods and Applications

Von der Fakultät für Mathematik und Informatik
der Universität Leipzig
angenommene

Dissertation

zur Erlangung des akademischen Grades

Doctor rerum naturalium
(Dr. rer. nat.)

im Fachgebiet
Informatik

Vorgelegt von
Dr. med. Dipl.-Bioinf. Steve Hoffmann
geboren am 18. Mai 1978 in Gifhorn

Die Annahme der Dissertation wurde empfohlen von:
1. Prof. Dr. Rolf Backofen, Albert-Ludwigs-Universität Freiburg
2. Prof. Dr. Peter F. Stadler, Universität Leipzig

Die Verleihung des akademischen Grades erfolgt mit Bestehen der Verteidigung am 17.09.2014 mit dem Gesamtprädikat

summa cum laude

Meinen Müttern

Was es brauchte

Sicher ist, dass ich die erste Nacht in Dresden sicher unter einer Brücke hätte verbringen müssen. In Leipzig gab es ein Bett, eine Fettbemme und gut zu trinken. Dazu Ideen, Stütze, aufrichtige Zuwendung, ein Zuhause. Auch dafür werde ich mich bei Peter und Petra nicht angemessen bedanken können.

Es brauchte Genome. Nukleotid für Nukleotid. Es brauchte Jens, der seine Hilfe nicht selten auf Kosten eigener Beduerfnisse anbot. Es brauchte Kollegen, Freunde, Familie. Anne, Christian, David, Gero, Frank und Helene waren die guten Gründe warum stets gerne in der Härtelstraße zu sein. Es brauchte einen Donath, einen Sven. Es brauchte Glück, eine PCR, Lesca und Daniel. Es brauchte das Beyerhaus, das Flowerpower, Nico, die unbekannte Frau. Meinen lieben Doktor Bernhart. Danke für die Lieder, Favoriten, die Geduldsamkeit, die Rezpete, die Karten, den Jägermeister, die guten und die schlechten Witze.

Se necescita mi Maribelita. Para la contradicción de todo lo que estoy acostumbrado.

Abstract

This thesis introduces three different algorithmical and statistical strategies for the analysis of high-throughput sequencing data. First, we introduce a heuristic method based on enhanced suffix arrays to map short sequences to larger reference genomes. The algorithm builds on the idea of an error-tolerant traversal of the suffix array for the reference genome in conjunction with the concept of matching statistics introduced by Chang and a bitvector based alignment algorithm proposed by Myers. In practice, for some suffix i of a read of length m, the suffix array is traversed until a mismatch occurs or all $m - i + 1$ characters of the suffix have been matched. In the latter case, the matching continues with the subsequent suffix $i + 1$. Otherwise, the algorithm evaluates alternative branches along the matching path of suffix i. The longest path will be selected as a seed alignment and the mapping continues with the suffix $i + 1$. The use of suffix link intervals helps to avoid remapping of characters that have already been matched in the traversal for suffix i. Note that a matching path does not necessarily end in a singleton interval, i.e. a leaf of the equivalent suffix tree. This implicitly allows multiple seed matches. In this way, it is possible to generate $m - \ell$ seed alignments in $O(m)$ time, where ℓ is a predefined minimum seed length. The seed alignments are subsequently used to trigger a number of bitvector based dynamic programming alignments. While this algorithm has quadratic complexity in theory, the encoding of characters as bitvectors leads to a linear runtime in practice. Based on user defined parameters either all read-reference alignments below a given error threshold or only the best read-reference alignments are reported. The algorithm supports paired-end and mate-pair alignments and the implementation offers methods for primer detection, primer and poly-A trimming. In our own benchmarks as well as independent benchmarks this tool outcompetes other currently available tools with respect to sensitivity and specificity in simulated and real data sets for a large number of sequencing protocols.

Second, we introduce a novel dynamic programming algorithm for the spliced alignment problem. The advantage of this algorithm is its capability to not only detect co-linear splice events, i.e. local splice events on the same genomic strand, but also circular and other non-collinear splice events. This succinct and simple algorithm handles all these cases at the same time with a high accuracy. While it is at par with other state-of-the-art methods for collinear splice events, it outcompetes other tools for many non-collinear splice events. The application of this method to publically available sequencing data led to the identification of a novel isoform of the tumor suppressor gene p53. Since this gene is one of the

best studied genes in the human genome, this finding is quite remarkable and suggests that the application of our algorithm could help to identify a plethora of novel isoforms and genes.

Third, we present a data adaptive method to call single nucleotide variations (SNVs) from aligned high-throughput sequencing reads. We demonstrate that our method based on empirical log-likelihoods automatically adjusts to the quality of a sequencing experiment and thus renders a "decision" on when to call an SNV. In our simulations this method is at par with current state-of-the-art tools.

Finally, we present biological results that have been obtained using the special features of the presented alignment algorithm. Specifically, the alignment algorithm was used to generate a genome wide map of the transcription start sites of the human pathogen *Helicobacter pylori*. The capability of our algorithm to accurately map very short reads proved helpful to identify the pRNA in this species. Moreover, the error tolerance of the mapper allowed the identification of a number of post-transcriptional modifications in RNAseq data. Furthermore, the capability to deal with multiple alignments helped to establish an important link between the long non-coding RNA (lncRNA) ANRIL, an evolutionarily young transcript, and equally young ALU-repeat families. Regardless of the importance of this RNA in the development of cardiovascular diseases, our finding supports a hypothesis that lncRNAs might exert their regulatory potential by binding to repetitive elements. If true, this would immediately imply that this often neglected part of the genome needed to be studied in much greater detail to understand recent evolutionary developments.

Zusammenfassung

Diese Arbeit stellt drei verschiedene algorithmische und statistische Strategien für die Analyse von Hochdurchsatz-Sequenzierungsdaten vor. Zuerst führen wir eine auf enhanced Suffixarrays basierende heuristische Methode ein, die kurze Sequenzen mit grossen Genomen aligniert. Die Methode basiert auf der Idee einer fehlertoleranten Traversierung eines Suffixarrays für Referenzgenome in Verbindung mit dem Konzept der Matching-Statistik von Chang und einem auf Bitvektoren basierenden Alignmentalgorithmus von Myers. Für ein Suffix i eines Reads der Länge m wird der Suffixarray durchlaufen, bis ein Mismatch zwischen Read und Referenz auftritt oder alle $m - i + 1$ Zeichen des Suffix gematcht worden sind. Im letzteren Fall wird die Traversierung gleich mit dem nachfolgenden Suffix $i + 1$ fortgesetzt. Andernfalls wertet der Algorithmus zunächst alternative Zweige des Pfades für das Suffix i aus. Der längste Pfad wird als *Seedalignment* ausgewählt und erst anschließend wird das Verfahren wird für das Suffix $i + 1$ fortgesetzt. Die Verwendung von Suffixlink-Intervallen hilft, den Vergleich von jenen Zeichen zu vermeiden, die bereits für Suffix i verglichen wurden. Ein Pfad muss nicht notwendigerweise in einem Singleton, d.h. in einem Blatt des äquivalenten Suffixbaumes enden. Damit erlaubt das hier vorgestellte Verfahren die schnelle Identifikation multipler Hits im Referenzgenom. Insgesamt ermöglicht der vorgestellte Algorithmus $m - \ell$ Seedalignments in $O(m)$ Zeit zu berechnen, wobei ℓ die benutzerdefinierte Mindestlänge eines Seedalignments ist. Die Seeds werden anschließend verwendet, um eine Reihe semiglobaler Alignments zu berechnen. Dabei kommt das auf Bitvektoren basierte Verfahren von Myers zum Einsatz. Während der Algorithmus in der Theorie eine quadratische Komplexität hat, führt die Kodierung der Zeichen als Bitvektoren zu einer linearen Laufzeit in der Praxis. Basierend auf vom Benutzer definierten Parametern werden entweder alle Read-Referenz Alignments unter einer bestimmten Fehlerschwelle oder nur die besten Read-Referenz Alignments ausgegeben. Die vorgestellte Methode unterstützt Paired-End und Mate-Pair Alignments, bietet Methoden zur Erkennung von Primersequenzen und zum trimmen von Poly-A-Signalen an. Auch in unabhängigen Benchmarks zeichnet sich das Verfahren durch hohe Sensitivität und Spezifität in simulierten und realen Datensätzen aus. Für eine große Anzahl von Sequenzierungsprotokollen erzielt es bessere Ergebnisse als andere bekannte Short-Read Alignmentprogramme.

Zweitens stellen wir einen auf dynamischer Programmierung basierenden Algorithmus für das *spliced alignment problem* vor. Der Vorteil dieses Algorithmus ist seine Fähigkeit, nicht nur kollineare Spleiß-Ereignisse, d.h. Spleiß-Ereignisse auf dem gleichen genomischen Strang,

sondern auch zirkuläre und andere nicht-kollineare Spleiß-Ereignisse zu identifizieren. Das Verfahren zeichnet sich durch eine hohe Genauigkeit aus: während es bei der Erkennung kollinearer Spleiß-Varianten vergleichbare Ergebnisse mit anderen Methoden erzielt, schlägt es die Wettbewerber mit Blick auf Sensitivität und Spezifität bei der Vorhersage nicht-kollinearer Spleißvarianten. Die Anwendung dieses Algorithmus führte zur Identifikation neuer Isoformen. In unserer Publikation berichten wir über eine neue Isoform des Tumorsuppressorgens p53. Da dieses Gen eines der am besten untersuchten Gene des menschlichen Genoms ist, könnte die Anwendung unseres Algorithmus helfen, eine Vielzahl weiterer Isoformen bei weniger prominenten Genen zu identifizieren.

Drittens stellen wir ein datenadaptives Modell zur Identifikation von Single Nucleotide Variations (SNVs) vor. In unserer Arbeit zeigen wir, dass sich unser auf empirischen log-likelihoods basierendes Modell automatisch an die Qualität der Sequenzierungsexperimente anpasst und eine "Entscheidung" darueber trifft, welche potentiellen Variationen als SNVs zu klassifizieren sind. In unseren Simulationen ist diese Methode auf Augenhöhe mit aktuell eingesetzten Verfahren.

Schließlich stellen wir eine Auswahl biologischer Ergebnisse vor, die mit den Besonderheiten der präsentierten Alignmentverfahren in Zusammenhang stehen. Insbesondere wurde der Alignmentalgorithmus für Hochdurchsatz-Sequenzierungsdaten verwendet, um eine genomweite Karte von Transkriptionsstartstellen des Krankheitserregers *Helicobacter pylori* zu erstellen. Die Fähigkeit unseres Algorithmus zum genauen Mapping auch sehr kurzer Sequenzen ermöglichte es, auch bisher nicht kartierte RNAs identifizieren. In einem weiteren Projekt erlaubte die Fehlertoleranz des Mappers die Identifizierung einer Reihe posttranskriptioneller Modifikationen. Die Fähigkeit des Alignmentverfahrens auch multiple Hits auszugeben, half eine wichtige Verbindung zwischen der evolutionär jungen langen nicht kodierenden RNA (lncRNA) ANRIL und den ebenso jungen ALU Repeats herzustellen. Unabhängig von der Bedeutung dieser RNA in der Pathogenese von Herz-Kreislauf-Erkrankungen, legen unsere Ergebnisse die Hypothese nah, dass lncRNAs ihr Regulierungspotenzial durch Bindung an repetitive Elemente entfalten. Würde sich diese Hypothese bestätigen lassen, wäre die genauere Untersuchung von repetitiven Elementen zum Verständnis rezenter evolutionäre Entwicklungen von besonderer Bedeutung.

Bibliographic Information

This Thesis is based on the following publications. Three first authorships, one joint first authorship, one last authorship, two second authorships and one fifth authorship

[Hoffmann 2009] **Steve Hoffmann**, Christian Otto, Stefan Kurtz, Cynthia M Sharma, Philipp Khaitovich, Jörg Vogel, Peter F Stadler and Jörg Hackermüller. Fast mapping of short sequences with mismatches, insertions and deletions using index structures. **PLoS Computational Biology**, vol. 5, page e1000502, 2009.

[Sharma 2010] Cynthia M Sharma, **Steve Hoffmann**, Fabien Darfeuille, Jéréemy Reignier, Sven Findeiß, Alexandra Sittka, Sandrine Chabas, Kristin Reiche, Jörg Hackermüller, Richard Reinhardt, Peter F Stadler and Jörg Vogel. The primary transcriptome of the major human pathogen Helicobacter pylori. **Nature**, vol. 464, no. 7286, pages 250-5, Mar 2010.

[Hoffmann 2011] **Steve Hoffmann**. Computational analysis of high throughput sequencing data. **Methods Mol Biol**, vol. 719, pages 199-217, 2011.

[Findeiss 2011] Sven Findeiß, David Langenberger, Peter F Stadler and **Steve Hoffmann**. Traces of post-transcriptional RNA modifications in deep sequencing data. **Biol Chem**, vol. 392, no. 4, pages 305-13, Apr 2011.

[Schmidtke 2012] Cornelius Schmidtke, Sven Findeiß, Cynthia M Sharma, Juliane Kuhfuß, **Steve Hoffmann**, Jörg Vogel, Peter F Stadler and Ulla Bonas. Genome-wide transcriptome analysis of the plant pathogen Xanthomonas identifies sRNAs with putative virulence functions. **Nucleic Acids Research**, vol. 40, no. 5, pages 2020-2031, 2012.

[Richter 2012] Julia Richter, Matthias Schlesner, **Steve Hoffmann***, Markus Kreuz, Ellen Leich, Birgit Burkhardt, Maciej Rosolowski, Ole Ammerpohl, Rabea Wagener, Stephan H. Bernhart, Dido Lenze, Monika Szczepanowski, Maren Paulsen, Simone Lipinski, Robert B. Russell, Sabine Adam-Klages, Gordana Apic, Alexander Claviez, Dirk Hasenclever, Volker Hovestadt, Nadine Hornig, Jan O. Korbel, Dieter Kube, David Langenberger, Chris Lawerenz, Jasmin Lisfeld, Katharina Meyer, Simone Picelli, Jordan Pischimarov, Bernhard Radlwimmer, Tobias Rausch, Marius Rohde, Markus Schilhabel, René Scholtysik, Rainer Spang, Heiko Trautmann, Thorsten Zenz, Arndt Borkhardt, Hans G. Drexler, Peter Möller, Roderick A.F. MacLeod, Christiane Pott, Stephan Schreiber, Lorenz Trümper, Markus Loeffler, Peter F. Stadler, Peter Lichter, Roland Eils, Ralf Küppers, Michael Hummel, Wolfram Klapper, Philip Rosenstiel, Andreas Rosenwald, Benedikt Brors and Reiner Siebert. Recurrent mutation of the ID3 gene in Burkitt Lymphoma identified by integrated genome, exome and transcriptome sequencing. **Nature Genetics**, vol. Vol. 44 (12), pages 1316-1320, 2012. * joint first authorship for Transcriptome Part.

[Holdt 2013] Lesca M. Holdt, **Steve Hoffmann**, Kristina Sass, David Langenberger, Markus Scholz, Knut Krohn, Knut Finstermeier, Anika Stahringer, Wolfgang Wilfert, Frank Beutner, Stephan Gielen, Gerhard Schuler, Gabor Gäbel, Hendrik Bergert, Ingo Bechmann, Peter F. Stadler, Joachim Thiery and Daniel Teupser. Alu Elements in ANRIL Non-Coding RNA at Chromosome 9p21 Modulate Atherogenic Cell Functions through Trans-Regulation of Gene Networks. PLoS Genetics, vol. 9, no. 7, page e1003588, 07 2013.

[Hoffmann 2014a] **Steve Hoffmann**, Christian Otto, Gero Doose, Andrea Tanzer, David Langenberger, Sabina Christ, Manfred Kunz, Lesca Holdt, Daniel Teupser, Jörg Hackermüller and Peter Stadler. A multi-split mapping algorithm for circular RNA, splicing, trans-splicing and fusion detection. **Genome Biology**, vol. 15, no. 2, page R34, 2014.

[Hoffmann 2014b] **Steve Hoffmann**, Peter F. Stadler, Korbinian Strimmer. A Simple Data-Adaptive Probabilistic Variant Calling Model. 2014. Submitted.

Contents

1 Introduction **1**

1.1 Towards the 1000-Dollar-Genome . . . 1

1.2 Sequencing platforms . . . 3

  1.2.1 454 pyrosequencing . . . 3

  1.2.2 Illumina . . . 4

  1.2.3 Helicos . . . 5

  1.2.4 SOLiD . . . 6

  1.2.5 Pacific Biosciences . . . 7

  1.2.6 Other approaches to sequencing . . . 7

  1.2.7 Paired-Ends . . . 8

1.3 Computational methods in short read analysis . . . 8

  1.3.1 Basic data analysis and evaluation . . . 8

  1.3.2 Mapping . . . 9

  1.3.3 Assembly of short read data . . . 11

  1.3.4 Variation calling . . . 12

2 Sequences and Alignments **15**

2.1 Sequences and Distances . . . 15

2.2 Local alignment . . . 20

2.3 Semi global alignment . . . 21

2.4 Expectation of a random string . . . 25

3 Suffix arrays **29**

3.1 The suffix trie and tree . . . 29

  3.1.1 Linear time construction of suffix trees . . . 32

3.2 Suffix arrays . . . 33

  3.2.1 Linear time construction of suffix arrays . . . 35

3.3 Enhanced suffix arrays . . . 38

  3.3.1 lcp-table . . . 39

  3.3.2 child table . . . 40

  3.3.3 suffix link intervals . . . 45

4 A heuristic method for short read alignment **47**

4.1 Outline of a novel mapping approach . . . 48

4.2 Methods . . . 52

4.3 Results . . . 63

5 Split read mapping **69**

5.1 Introduction . . . 69

5.2 An algorithm for split read mapping . . . 71

5.3 Results . . . 75

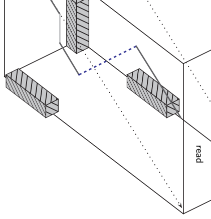
6 A data adaptive model for variation detection **89**
6.1 Introduction 89
6.2 The statistical model 92
6.3 Results 96

7 Biological applications **101**
7.1 The primary transcriptome of *H. pylori* 101
7.1.1 dRNAseq of the *H.pylori* transcriptome 102
7.1.2 Mapping and basic sequence analysis 103
7.1.3 Identifcation of start sites 105
7.1.4 Differential expression analysis 109
7.2 An insight to post-transcriptional modification 111
7.2.1 Short RNA sequence data 111
7.2.2 Chemical modifications 113
7.2.3 Non-templated CCA-additions 115
7.3 Mutated splice sites in Lymphoma samples 117
7.4 Short read alignments to repetitive regions 119

8 Discussion **121**

A Appendix **127**

Bibliography **129**



Chapter 1

Introduction

Contents

1.1 Towards the 1000-Dollar-Genome 1

1.2 Sequencing platforms 3

  1.2.1 454 pyrosequencing 3

  1.2.2 Illumina 4

  1.2.3 Helicos 5

  1.2.4 SOLiD 6

  1.2.5 Pacific Biosciences 7

  1.2.6 Other approaches to sequencing 7

  1.2.7 Paired-Ends 8

1.3 Computational methods in short read analysis 8

  1.3.1 Basic data analysis and evaluation 8

  1.3.2 Mapping 9

  1.3.3 Assembly of short read data 11

  1.3.4 Variation calling 12

This chapter introduces the technological, biological and computational hallmarks of High Throughput Sequencing (HTS). It summarizes shortcommings, advantages and potential applications in this field. The chapter largely follows [Hoffmann 2011].

1.1 Towards the 1000-Dollar-Genome

The advent of High Throughput Sequencing (HTS) methods opens new opportunities for the analysis of genomes and transcriptomes. While the sequencing of a whole mammalian genome took several years at the turn of this century, today it is only a matter of weeks. The race towards the thousand-dollar genome is fueled by the - ethically challenging - idea of personalized genomic medicine. However, these methods allow new and interesting insights into many aspects of life such as the discovery of novel non-coding RNA classes, structural variants or alternative splice sites to name a few. Meanwhile, several methods for HTS have been introduced to the markets. Here a short overview on the technologies and the bioinformatic analysis of HTS data is given. Turning to the High Throughput Sequencing (HTS), often also referred to as Next Generation Sequencing (NGS), one quickly realizes that the time of spreadsheet-bioinformatics has finally come to an end. A single 3-day sequencer run confronts the scientists with terabytes of data: images, qualities, statistics, summaries, sequences, maps and assemblies to name few. In the light of this quick and massive avalanche of data, deciding on data storage policies alone appears to be a difficult task. The deletion of any file will inevitably limit the possibilities of reanalysis. Especially for precious biological samples it might be worthwhile to store the data. For example, the reanalysis of sequencer camera images with alternative base calling tools may yield important improvements. One of the major vendors of sequencing machines, Illumina Inc., has announced to ship the first sequencing machines, the HiSeq X Ten, that is able to sequence whole human genomes at 30-fold coverage for approximately 1000 US dollars [Illumina Inc. 2014]. This technology however is going to be limited to sequencing the human genome.

Because the prices for whole-genome sequencing of mammalian genomes have dropped significantly [Pushkarev 2009] and the time needed for sequencing does not exceed a couple of weeks, yet unknown amounts of data will accumulate in laboratories all over the world. Hence, whatever policies are adopted, and irrespective of whether the sequencing itself is outsourced or not, a series of smaller HTS projects already requires a specialized infrastructure with arrays of hard disks, fast network architectures or even tape storages. These issues may not be overstressed since they heavily affect everyday work with the HTS data. A few questions have to be answered carefully in each case. How to exchange data with the collaborators? How to set up pipeplines for analysis? Is a version control of reanalyzed data needed? Picturing a tape storage library with loads of TB-cartridges in a molecular biology lab one immediately realizes that the analysis of HTS data is a problem of its own. Many standard algorithms and tools frequently used in genome informatics had or still have to be revised in order to contain the deluge of HTS data. Moreover, HTS offers the opportunity for new types of analysis such as transcription start site detection, ncRNA detection or RNA structure probing requiring new algorithms and standards. This chapter intends to give an overview on the sequencing technologies as well as basic approaches to HTS data analysis. As the HTS methods are steadily improving and changing at a very high pace, the focus will be on their basic principles rather than fugacious facts or specific pros and cons of single technologies. Despite their different error models, properties and application areas, all HTS approaches gain their true beauty and fascination by their genuine combination of different techniques from various fields of science: molecular biology, chemistry, physics, material science, engineering and computer science.

Current high throughput DNA sequencing methods may be subdivided in two major approaches: sequencing by ligation and sequencing by synthesis. In sequencing by synthesis a single stranded, primer probed DNA template is sequentially duplicated by a polymerase. During duplication chemically modified nucleotides allow the detection of each base of the template. On the other hand, sequencing by ligation does not use polymerases but employs specifically binding primer probes. ABI's SOLiD sequencing platform [Pandey 2008] puts this idea into practice. However, in the near future a direct read out of sequences using the physico-chemical properties of nucleotides (such as charges) may become the state of the art.

1.2 Sequencing platforms

In principle all novel sequencing methods achieve high throughput by immobilizing large amounts of single DNA or cDNA fragments locally. Regardless, whether the immobilization takes place on beads or plane surfaces - the idea is to separate fragments sufficiently to perform the sequencing for all fragments at once. Meanwhile, several next-generation sequencing technologies introduced in the second half of the 2000's have been announced to be discontiued. One manufacturer of a next-generation sequencing technology declared bankrupcy (Helicos Biosciences) in late 2012 [GenomeWeb 2012]. A remarkable exception is the technology brought forward by Pacific Biosystems.

1.2.1 454 pyrosequencing

The 454 pyrosequencing system was the first HTS system introduced to the markets [Margulies 2005]. The key idea of pyrosequencing is to trigger detectable chemiluminescent reactions during the sequencing step. To prepare a DNA library for the 454 platform double stranded source material needs to be sheared into fragments of several hundred nucleotides. In a second step two different linkers, i.e. specific DNA sequences of known length and composition, are ligated to the fragments. A 5'-biotin tag attached to one of the linkers allows the immobilization of the fragments on DNA capture beads. An excess of beads ensures that the expected number of bound fragments per bead does not exceed one. To generate detectable light signals during the synthesis step an emulsion polymerase chain reaction (emPCR) clonally amplifies the fragment. During this process millions of copies are directly attached to the bead. Finally, the beads are locally immobilized. Picotiter plates, i.e. plates with millions of microscopically small wells just large enough to contain a single bead, are used to trap the beads. The nucleotides adenine (A), cytosine (C), thymine

Technology	Read length	bp per run	Accuracy[1]	Run time	Remarks[2]
Roche 454 (Titanium)	~400 bp	~400MB	99.5%	7h	Supports mate pair indels in homopolymers
Illumina (HiSeq 2000)	36-100 bp	~95-600 GB	>98%	2-11 days	Supports paired end and mate pair
SOLiD (5500 W)	~50 bp	~80-320 GB	99.94%	14 days	Supports paired end
PacBio RS II	~1-30 kbp	~375 Mbp	90%	3h	Vendor claims “consensus” accuracy of >99.999% for coverages >30.

Table 1.1: Comparison of high-throughput sequencing methods

(T) and guanine (G) are sequentially washed over the plates in four recurring cycles. Polymerases along with other enzymes generate light signals when the nucleotides are incorporated into the DNA strand. More precisely, the enzyme luciferase is the major component of the chemiluminescent reaction (Fig. 1.1). After each cycle, a washing step is needed to remove the excess of nucleotides from the plate. The description of the sequencing by synthesis step already reveals an important problem: sequencing of homopolymers. A stretch of two or more identical nucleotides in the DNA template will generate multiple subsequent non-synchronous chemi-luminescent reactions during a single cycle. Hence, the intensity of the light signal is the only way to determine the length of a homopolymer - making the use of rather complicated signal processing steps necessary. Additionally, despite the washing steps, nucleotides accumulate within the wells causing an increase of the background signal as the sequencing proceeds. The major advantage of the 454 method in comparison with other technologies is the longer read length making it sequences especially useful for assemblies (Tab. 1.1). In late 2013 Roche has announced the shutdown of the 454 sequencing technology [GenomeWeb 2013] in favor of a joint venture with Pacific Biosystems. The sequencers will be phased out in 2016.

1.2.2 Illumina

With only up to 100 nucleotides, Illumina fragments are significantly shorter compared to those from 454 [Bentley 2008]. However, due to its degree of parallelization this method allows a higher throughput with up to 20 gigabases per run (cf. Tab. 1.1). In contrast to 454, fragments ligated to two different adapters are immobilized on translucent plates, flow cells, densely coated with oligonucleotides complementary to the adapters. The key idea of this procedure is to clonally amplify the fragments in a circumscribed region of the flow cell. The complementary adapters on the plate act as PCR primers to generate clusters of millions of identical copies using a process called bridge-amplification. The prerequisite for a successful generation and detection of such clusters is that the initially binding fragments are well separated from each other (Fig. 1.2). In fact, separa-

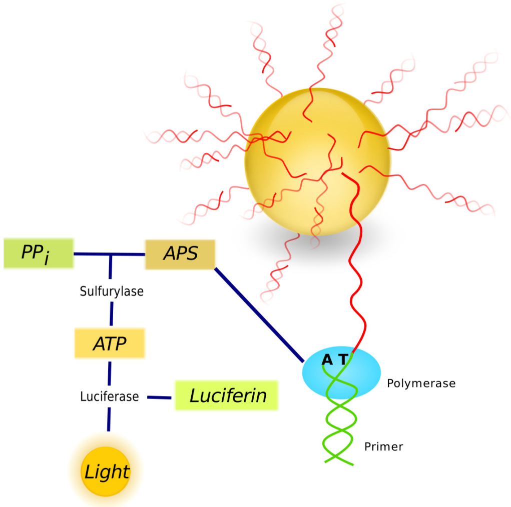


Figure 1.1: 454 Pyrosequencing. After clonal amplification of DNA fragments at DNA capture beads, beads are trapped in small wells of a picotiter Plate. During the sequencing-by-synthesis the four nucleotides are washed in recurring cycles over the plate. Incorporation of nucleotides by the polymerase results in the production of ATP. In turn, the energy rich ATP triggers a luciferase reaction generating a light signal recorded by a CCD camera. 454 Sequencing (c) Roche Diagnostics. All rights reserved.

bility and purity of clusters is one of the major challenges in the signal detection step. The sequencing by synthesis step involves reversible dye-terminators. The polymerase incorporates differently labeled nucleotides that bring the sequencing to a sudden stop. Laser excitation of labels indicates the type of the incorporated nucleotide and cleaves the terminator activity. Subsequently, the next base can be called. The Illumina approach intends to call bases synchronously, i.e. all light signals generated in the cycle belong to the nucleotide of all fragments in the flow cell. In practice, however, this does not always work. A failure to remove the terminator or the integration of nucleotides without a terminator activity, e.g., will result in a phase shift. This phasing increases noise and complicates the signal detection.

1.2.3 Helicos

Helicos has styled its single molecule sequencing figuratively as DNA microscopy. Although the company's own interpretation was at least semantically questionable (a microscope typically does not evaluate optical data) their system provided a solution that did not require amplification, introducing additional biases [Pushkarev 2009, Harris 2009]. Instead of specific adapter sequences, poly-A anchors were covalently attached to single fragments. Subsequently, glass cover slips covered with poly-T oligomers were used to capture the library fragments. During sequencing the plate is incubated with only one Cy5-labled nucleotide at a time and a light sig-

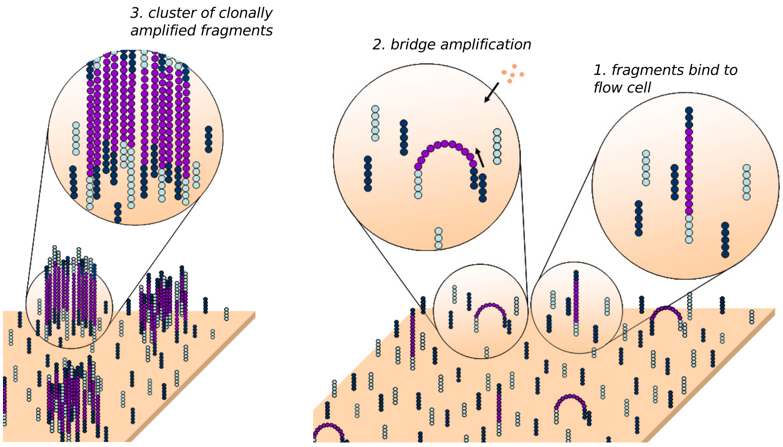

3. cluster of clonally amplified fragments
2. bridge amplification
1. fragments bind to flow cell

Figure 1.2: Illumina/Solexa cluster generation. Adapter-tagged fragments are immobilized on glass cover slips densely coated with reverse complementary adapters (1). Subsequently, fragments are amplified using a bridge amplification step (2), resulting in locally separated clusters of clonally amplified fragments (3). Separability and purity of clusters is a key prerequisite for the Illumina technology. During the sequencing step (using reversible terminator dye chemistry) the clusters generate sufficiently strong light signals to be detected by a camera. (c)2010, Illumina, Inc. All rights reserved.

nal is generated upon laser excitation. Because of the low signal strength good background discrimination and high-resolution image detection was required. Indeed, the error-rates of Helicos are reported to be significantly higher in comparison with other HTS technologies [Margulies 2005] , demanding more sensitive software in the downstream analysis (cf. Tab. 1.1). Meanwhile, Helicos has declared bankrupcy [GenomeWeb 2012].

1.2.4 SOLiD

Life technologies' SOLiD technology has employed a sequencing by ligation approach. Sheared fragments coupled to an universal primer (P1) are attached to beads and clonally amplified in an emPCR reaction. In contrast to 454, beads are not captured in wells but covalently bound to glass plates. In the first sequencing round a universal primer complementary to the 3′-end of P1 is used. Subsequently, fluorescently labeled 5mer-probes are washed over the plate. Each label represents a set of four dinucleotide combinations specific only for the first and second position of the probe, while the other three nucleotides are random. Probes complementarily binding to the template sequence are ligated to the primer. After signal detection the labels are simultaneously cleaved and a second ligation reaction elongates the extension product. At the end of the first ligation round with multiple ligation reactions the color information is obtained for pairs of nucleotides with a lag of 4nt, i.e. the 1st and 2nd, the 6th and 7th, the 11th and the 12th and so one. Note that the color information from the first round is not sufficient to decode the bases. Hence, additional ligation rounds are necessary to decode the colors and to interrogate the remaining nucleotides. Therefore, the extension product is removed and a second primer complementary to the 3′-end of P1 with an offset of -1 binds to the template. In this ligation round the first two interrogated nucleotides are the last base of P1 and the first base of the template. With the color codes obtained during the first and the second run, the nucleotides 1, 6, 11, 12 etc. can be decoded. In total, it requires 5 ligation rounds to decode the whole template. Due to the fact that each base is interrogated twice, a higher accuracy is expected. Another sequencing by ligation approach called combinatorial probe anchor ligation (cPAL) is used by Complete Genomics. The company currently offers sequencing services only. Complete Genomics was the first to announce the sequencing of a whole human genome with a coverage of 78-fold for less than 5000 US$ [Drmanac 2009]. Meanwhile, the company has developed a bead-free protocol [Life technologies 2012]

1.2.5 Pacific Biosciences

A different technology, currently developed by Pacific Biosciences locally immobilizes the polymerases on a glass surface. Each polymerase is surrounded by a zero-mode waveguide (ZMW) that confines light in volumes (typically <10e-21 liters) smaller than its wavelength [Eid 2009]. Using ZMWs light signals generated upon the integration of labeled nucleotides can be distinguished from the background. While the reads are reported to be significantly longer, error rates are reported to be significantly higher, especially at their ends (personal communication by Chris Ameniya 2013). Thus, more sophisticated algorithms are required for the mapping and denovo assembly using these reads. The major advantage, read lengths of up to 20000 nt, is reported to be especially useful to fill gaps in current genome assemblies [English 2012, Huddleston 2014]. Meanwhile, smaller genomes such as the genome a of thyphimurium serovar of Salmonella enterica [Hoffmann 2013] have been published using the PacBio technolgy.

1.2.6 Other approaches to sequencing

Nanopore sequencing is an alternative approach to HTS. The initial concept was based on threading a DNA molecule through a staphylococcal alpha-haemolysine. During passage, the DNA alters the current applied to the alpha-haemolysine. These modulations are specific for the nucleotide composition. Newer publications investigate the usage of covalently incorporated adapters to detect single nucleotides cleaved from DNA fragments upon exonuclease treatment by means of current modulations [Clarke 2009]. Under optimal conditions this approach was reported to achieve acceptable error rates.

1.2.7 Paired-Ends

The sequencing of paired-ends helps to overcome the problem of ambiguity caused by short read lengths and improves the assembly results. Using DNA circularization during the library construction, shorter fragments can be obtained from both ends of a larger genomic fragment with a typical length of 2-5kb. The pair of small DNA fragments is then sequenced in the same cluster or well. The prior knowledge of the approximate distance between the two fragments impedes the misplacement of both reads during alignment and helps to assemble repetitive regions. Furthermore, it is useful to detect structural variations such as copy number variations. Note that there is some confusion about the terminology. In the Illumina protocol the process described above is termed mate-pair sequencing, while the 454-protocol refers to it as paired-end sequencing. Illumina's paired-end sequencing does not require DNA circularization: a little modification of the single-end protocol allows the sequencing of a DNA fragment of about 200-500bp from both ends simultaneously.

1.3 Computational methods in short read analysis

As mentioned above, the sheer amount of data demand efficient and fast algorithms for the bioinformatic analysis. In principle, the analysis already begins with the base calling, i.e. the decoding of a time series of electromagnetic signals to the nucleotide sequences and associated quality values. Meanwhile, several independent groups have proposed alternative base calling approaches with optimized results [Quinlan 2008, Kircher 2009, Erlich 2008]. These concepts vary from bayesian base calling methods to approaches using support vector machines. Here, we only focus on some standard input and output formats as well as the two most common forms of sequence data analysis: mapping and assembly.

1.3.1 Basic data analysis and evaluation

For further downstream analysis base callers typically report the sequences together with their quality values. While the Illumina platform reports the sequences along with the quality values in a multiple FASTQ format, reads and quality values from the 454 GS FLX are often exported from the binary SFF format to two separate multiple FASTA files. However, the Genome Sequencer Data Analysis suite offers several tools to process the SFF files directly. For 454 reads, the header line starts with a unique identifier followed by the read length, the coordinates of the bead and the date of the sequencing run. For Illumina reads, the header informs about the name of the instrument , the flowcell lane, the tile number within the flow cell, its x- and y-coordinates and has a flag indicating whether the read is single-end or belongs to a mate-pair or paired-end run. Note that the qualities are given in ASCII code. The quality values give an estimate on the accuracy of the base calling. Nowadays most sequencing platforms report a Phred quality score. The score, originally developed in the context of the Human Genome project, is given by

$$Q = -10 \cdot \log_{10} p \tag{1.1}$$

where p is the probability that the reported base is incorrect. Illumina initially decided to deviate from this scoring and instead used the formula

$$Q_{\text{Illumina}} = -10 \cdot \log_{10} \frac{p}{1-p} \tag{1.2}$$

While the Illumina quality score is asymptotically identical to Q for low error probabilities, it is typically smaller for higher error probabilities. Since the Solexa quality scores can become negative, a conversion to real phred scores using

$$Q = 10 \cdot \log_{10}(1 + 10^{Q_{\text{Solexa}}/10}) \tag{1.3}$$

may be necessary. While high Illumina quality scores have been reported to overestimate the base calling accuracy, low scores underestimated the base calling accuracy [Dohm 2008, Kircher 2009]. Since version 1.3 the proprietary Solexa pipeline uses Phred scores. It is important to note that also the encoding of the quality string has been subject to changes. The new pipeline encodes the phred qualities from 0 to 62 in a non-standard way using the ASCII characters 64 to 126. Due to the fact that Phred scores from 0 to 93 are normally encoded using ASCII characters 33 to 126, a conversion might be necessary.

1.3.2 Mapping

In genome informatics *mapping* describes the process of generating a (mostly heuristic) alignment of query sequences to reference genomes. It is the basis for qualitative as well as quantitative analysis of HTS data. The mapping may be used to identify genomic regions that encoded for the query sequences, to find single nucleotide polymorphisms or to estimate the expression level of a certain transcript. Compared to sequences obtained by Sanger sequencing, the algorithms currently have to address three different problems at once. In addition to the tremendous amounts of data, the methods have to deal with a lower data quality and shorter read lengths. Especially for short (erroneous) reads it is often not possible to decide for a position of origin in the reference genome because the reads align equally well to several genomic locations. Sequencing of repetitive regions complicates this problem even more. The applications presented here apply different mapping policies to tackle those problems. To address the problem of the huge amount of data, most of the short read alignment programs use index structures either for the reads or the reference. A recent survey of the EBI counted more than 80 next generation sequencing mappers [Fonseca 2012] (retrieved April 2014). The majority of these mappers in principle share the same concepts and mapping strategies. These concepts, along with the tools that used them first, are summarized below

1.3.2.1 Mapping with hash tables

Heng Li et al. developed one of the first read mappers, MAQ, for Illumina sequences based on hash tables. Although the tool is no longer supported, a look into the core of this approach reveals some basic principles and policies of short read mapping. The focus of MAQ is to incorporate quality values to facilitate and improve read mapping [Li 2008a]. By default, MAQ indexes only the first 28bp of the reads (the seed) in six different hash tables ensuring that all reads with at most 2 mismatches may be found in the genome. Assuming a read of only 8bp the hash tables are build from 3 pairs of complementary templates ,11110000, 00001111, 11000011, 00111100, 11001100, and 00110011, where a 1 indicates a base that is included in the hash key generation. After this indexing step, MAQ proceeds by scanning the reference sequence once for each pair of complementary template sequences. Each time a seed hit is encountered MAQ attempts to extend the hit beyond the seed and scores it according to the quality values. It has been reported earlier that the use of quality values during the read alignment can improve the mapping results substantially [Smith 2008]. By default MAQ reports all hits with up to 2 mismatches - but only 57% with three mismatches. Hits with insertions and deletions (indels) are not reported. Furthermore, for reads with multiple equally scoring best hits only one hit is reported.

1.3.2.2 Mapping with suffix arrays and the Burrows-Wheeler Transform

A second approach to short read alignment uses the Burrows-Wheeler Transform (BWT). In brief, the BWT is a sorted cyclic permutation of some text T, e.g. a reference genome. Its main advantage is that the BWT of T contains stretches of repetitive symbols - making the compression of T more effective. The backward search algorithm [Ferragina 2000] on a compressed BWT simulates a fast traversal of a prefix trie for T - without explicitly representing the tree in the memory. It only requires two arrays to efficiently access the compressed BWT, which is the key to the speed and the low memory footprint of read aligners such as BWA [Li 2009a], Bowtie [Langmead 2009] and SOAP2 [Li 2009d]. Because the backward search only finds exact matches, additional algorithms for inexact searches had to be devised. BWA, for example, solves this problem by enumerating alternative nucleotides to find mismatches, insertions and deletions, while SOAP2 employs a split alignment strategy. Here, the read is split into two parts, to allow a single mismatch, insertion or deletion. The mismatch can exist in at most one of the two fragments at the same time. Likewise, the read is split into three fragments to allow two mismatches and so forth. Other tools such as Bowtie do not allow short read alignments with gaps. BWT based read mappers are the speed champions of short aligners - with an exceptionally low memory footprint. However, for all of the tools described above, the user has to carefully choose a threshold for a maximum number of acceptable errors. For error thresholds >2 mismatches, insertions or deletions the speed decreases significantly. While these thresholds seem to be sufficient for mapping of genomic DNA data, mapping of transcriptome data or data that contains contaminations (e.g. linkers) may prove difficult.

In contrast the tool segemehl presented in this work [Hoffmann 2009, Hoffmann 2014a], based on enhanced suffix arrays, aims to find a best local alignment with increased sensitivity. In a first step, exact matches of all substrings of a read and the reference genome are computed. The exact substring matches are then modified by a limited number of mismatches, insertions and deletions. The set of exact and inexact substring matches is subsequently evaluated using a fast accurate alignment method. While the program shows good recall rates of 80% for high error rates around 10%, it has a significantly larger memory footprint in comparison with the BWT and hashing methods.

The selection of an appropriate mapping method depends on various criteria. Due to the different indexing techniques some short read aligners are limited to certain read lengths. These tools may not be used if long reads or reads of different sizes need to be aligned. Furthermore, for speed reasons some aligners report only one hit per read - regardless of whether multiple equally good hits are obtained. This may be a problem if repetitive regions are sequenced. The user has to carefully assess which degree of sensitivity is needed: while a method that discards reads with multiple hits (sometimes a random hit is reported) or high error rates is suitable for SNP detection, mapping of transcriptome (RNAseq) data may require a higher sensitivity.

1.3.3 Assembly of short read data

The advent of HTS has raised hopes to quickly and inexpensively perform de-novo assemblies of large genomes. However, shorter read lengths and higher error rates have spoiled all too optimistic expectations. One of the first tools for short read assembly, SSAKE [Warren 2007], employs a greedy method to build larger contigs from short Illumina reads. After building a hash table holding unique read sequences, a prefix tree indexes all such sequences. The assembly starts with the most abundant unique sequence. All 3'most k-mers, i.e. substrings of length k with a distances of at most m the 3', are looked up in the prefix tree. All hits are used to build the first consensus contig. This consensus is then used to find the next set of k-mers. This process is iterated until all possibilities of the contig extension are exhausted. While such a simple method works well for small genomes, the assembly of larger genomes from short reads is rather cumbersome. Zerbino et al. [Zerbino 2008] used a more complicated deBruijn-Graph approach. Each node in the graph represents a series of overlapping k-mers, such that two adjacent k-mers overlap by k-1 characters. Two nodes A and B are connected by a directed edge if the last k-mer of the node A overlaps with the first of B. Hence, not only the reads but a whole series of overlapping reads can be modeled as a path through the graph. The authors report that their program Velvet is capable of assembling bacterial genomes such that half of the genomes is assembled to contigs of 50 kb or more (N50 measure). In simulations with 5-Mb regions of large mammalian genomes, the N50 contig length was up to 3 kb long. If available, both applications make use of mate-pair information. In the future, alternative approaches that combine the high coverage provided by short read sequencers such as Illumina with longer 454 and Sanger reads may prove to be more effective when it comes to the assembly of larger vertebrate genomes. However, another tool that employs deBruijn-Graphs, SOAPdenovo, was used to successfully assemble mammalian genomes from single-end and mate-pair Illumina sequences only [Li 2010c, Li 2010b].

1.3.4 Variation calling

The most important goal of personalized genomics is the detection of variations such as single nucleotide polymorphisms (SNP) and somatic variants, i.e. mutations. HTS offeres the opportunity to detect SNPs with minor allelic variants or tumor mutations on a large scale. Therefore, some vendors of HTS platforms such as Illumina provide tools to call SNPs directly from the sequencing data. The first alternative methods for SNP calling are, for example, offered by Samtools [Li 2009b] (cf. [Li 2008a]) or SOAPsnp [Li 2009c]. Meanwhile, the number of SNV callers and mutation callers has largely increased. The Genome Analysis Toolkit [McKenna 2010, DePristo 2011, Van der Auwera 2013] is one of the largest software suite for calling somatic and germline SNVs and is used in a number of large sequencing projects. Unfortunately, currently available SNV callers like GATK and SOAPsnp lack transpaency with respect to statistical models or are only available closed source. In fact, recent surveys indicate that the different somatic SNV calling models and pipelines, applied to the same data sets, show strikingly little overlap [Roberts 2013, Wang 2013, Li 2014]. Also for this reason it is necessary to openly publish the underlying SNV calling models more transparently and to discuss the influence of its variables. Some key ideas of currently used SNV calling models are discussed in the following.

Primary prerequisite for the SNP calling is a high quality of the sequencing data. To assure this, typically all reads with multiple hits, reads with low overall qualities and reads with more than one mismatch are discarded. The success of SNP calling in HTS data depends not only on the quality of the reads but also on the coverage. After mapping the reads to a reference, the cross section at sufficiently covered (e.g. >10) positions is checked for polymorphisms. To do this many SNP callers employ Bayesian statistics. For example, SNPsoap assumes a set of ten different genotypes

$$T = H_m H_n \in \{AA, CC, GG, TT, AC, AG, AT, CG, CT, GT\} \tag{1.4}$$

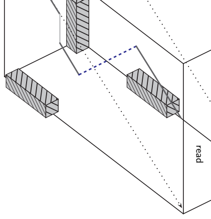
where H_m and H_n denote the two haplotypes of the genotype T_i at some position i of the genome. Typically, cross section data only contains the number of called bases along with their quality values. To obtain an estimate on the conditional probability of the data one may calculate

$$P(D \mid T) = \Pi_{i=1}^{l} \frac{P(d_k \mid H_m) + P(d_k \mid H_n)}{2} \tag{1.5}$$

where l is the number of observed nucleotides in the cross section (i.e. coverage) and $P(d_k \mid H)$ is the probability of observing the nucleotide d_k under the hypothesis H. The posterior probability for a genotype T_i given the HTS data D is then evaluated with

$$P(T \mid D) = \frac{P(T)P(D \mid T)}{\sum_{x=1}^{10} P(T_x)P(D \mid T_x)} \tag{1.6}$$

where the probability of a genotype $P(T)$ is usually calculated using prior knowledge. To reduce false-positive SNP calling SNPsoap additionally considers quality values. A similar approach was chosen for the Helicos pipeline [Harris 2009]. However, an important drawback of the approach sketched above is that the successful base calling depends on an equally successful coverage of both haplotypes and it may only be used for single individuals but not for pooled samples. It furthermore assumes that only two nucleotides segregate per site so that autosomal mutations may be missed. A maximum-likelihood method for analysing pooled samples was published by Michael Lynch [Lynch 2009].



Chapter 2

Sequences and Alignments

Contents

2.1 Sequences and Distances 15
2.2 Local alignment 20
2.3 Semi global alignment 21
2.4 Expectation of a random string 25

Sequences are concatenations of symbols. These symbols may be numbers or other elements such as characters drawn from the latin alphabet. Whether in Newspapers, telephone books, MP3 files, in Biology, in Mathematics - sequences are used in virtually all aspects of Life to store information. In essence, the recovery, comparison and understanding of this serialized information is at the very heart of this thesis. A special focus is put on DNA molecules, sequences over a small alphabet of only four charaters. In this chapter sequences over given alphabets are informally and formally introduced. It provides a basis for the considerations and observations below.

2.1 Sequences and Distances

First, we observe that sequences are concatenations of symbols or characters drawn from an alphabet. The alphabet Σ is a finte set of different characters. For example, the Σ may represent the latin alphabet, the DNA alphabet or the protein alphabet. The set Σ provides a basis for a sequence S and thus we refer to S *over* the alphabet Σ. One example is the DNA alphabet $\Sigma_{\mathrm{DNA}} = \{A, C, G, T\}$ consiting of four letters, one for each of the DNA nucleotides Adenine, Cytosine, Guanine, and Thymine. In the following the words string and sequence will be used synomously. To proceed formally the emtpy string ε is introduced. In the following the term empty string will be synonymously interchanged with the expression empty character.

Definition 1 (set of finite strings over Σ) *The set of all strings Σ^*, called the free monoid of Σ, over a given finite alphabet Σ is given by the Kleene* *star [Hopcroft 2006] on* Σ *denoted by*

$$\Sigma^* = \bigcup_{i \geqslant 0} \Sigma_i \tag{2.1}$$

where $\Sigma_0 = \varepsilon$ *and* $\Sigma_{i+1} = \{aw \mid a \in \Sigma, w \in \Sigma_i\}$.

This recursive definition builds up the set of all sequences over the Alphabet by iterating over their length. The length of the sequence s is denoted by $|s|$. The set of non-empty strings over $Alphabet$ is denoted by Σ^+. To further analyse sequences it is sometimes necessary to partion the sequences.

Definition 2 (prefix, substring, suffix) *Assuming a sequence of the form* $s = uvw$ *with* $u, v, w \in \Sigma^*$ *we call*

- u *a prefix of* s
- v *an infix or substring of* s
- w *a suffix of* s

It is easy to see that every sequence may be partioned in this way since u, v, w may represent the empty string. Likewise, s is a prefix, a substring and a suffix of itself. The i-th character in s is denoted by $s[i]$. Suppose that S is a sequence of length $|S| = n$. We index S from position 0. That is, $S[i]$ denotes the character at position i in S, for $0 \leqslant i \leqslant n-1$. For $i \leqslant j$, $S[i..j]$ denotes the substring of S starting with the character at position i and ending with the character at position j. For $i > j$, $S[i..j]$ denotes the empty string.

When working with sequences several different questions may arise. In many cases, eg. searches for names in electronic phone books, it is necessary to perform an exact text matching. In cases where the spelling of the name is unknown the search needs to retrieve also inexact matches. In biology, it is often necessary to find all occurences of smaller query sequences, exact or inexact, in a large genome. Moreover, to analyze the phylogeny of multiple species it is often necessary to compare their genomes and use the genomic difference as a measure of their relationship. In essence, all these questions are about the evaluation of the degree of identity of sequences and finding an appropriate measure.

Definition 3 (metric) *A function* $d : \Sigma^* x \Sigma^* \to \mathbb{R}$ *that evaluates the identity of sequences over* Σ *is called metric if for all* $x, y, z \in \Sigma^*$ *the following properties are fullfilled:*

1. $d(x, y) \geqslant 0$ *(positivity)*
2. $d(x, y) = 0$ *iff* $x = y$ *(zero property)*

3. $d(x, y) = d(y, x)$ *(symmetry)*

4. $d(x, y) \leqslant d(x, z) + d(z, y)$ *(triangle inequality)*

However, in several cases of sequence comparison these strict properties may not be fullfilled. Reducing the zero property to $d(x, x) = 0$ results in d being a pseudo-metric. A very simple metric on strings is the *Hamming distance* of two sequences s_1 and s_2 of equal length $|s_1| = |s_2| = n$

$$d(s_1, s_2) = \sum_{i=0}^{n-1} (s_1[i] \neq s_2[i]) \tag{2.2}$$

where $(s_1[i] \neq s_2[i]) = 1$ iff $s_1 \neq s_2$ and 0 otherwise. Thus, it is obtained by simply summing up the mismatches of the two sequences.

A distance for sequences of different length

A more complex and more commonly used metric on sequences is the *Levenshtein distance* or *unit edit distance* or sometimes simply *edit distance* [Levenshtein 1966]. It does not require the two strings to be of the same length. To obtain the Levenshtein distance of two sequences of different length a *pairwise alignment* with substitutions, insertions and/or deletions is required. In principle an alignment is a series of edit operations (substitutions, insertions and deletions) to transform one sequence into the other. Briefly, the Levenshtein distance is the minimum number of non-matching operations needed to convert one sequence into the other. The formal introduction of alignments is given along the lines of [Kurtz 2006].

Definition 4 (edit operation) *A character edit operation is a character transition*

$$\alpha \to \beta, \alpha, \beta \in \{\Sigma \cup \varepsilon\} \tag{2.3}$$

where $\varepsilon \to \varepsilon$ is declared the empty edit operation that has no effect and is explicitly excluded.

It is intuitive to see that in case $a, b \in \Sigma$, ie. both symbols are non-empty characters, $a \to b$ denotes a *substitution or replacement*. This formal terminology is confusing since the case $a = b$ is explicitly allowed and thus the transition is not a true exchange of characters. However, to be in line with the literature in the field the terminology is adopted nevertheless. Furthermore $a \to \varepsilon$ denotes a deletion of the character a, $\varepsilon \to b$ the insertion of character a. Along the lines of the informal introduction of alignments given above we now introduce

Definition 5 (pairwise alignment) *A pairwise alignment* $\mathcal{A}(s_1, s_2)$ *of two sequences* s_1, s_2 *can be expressed as a series of edit operations.*

$$\mathcal{A}(s_1, s_2) = (\alpha_0 \rightarrow \beta_0, \cdots, \alpha_{h-1} \rightarrow \beta_{h-1}) \tag{2.4}$$

where $s_1 = \alpha_0 \cdots \alpha_{h-1}$ *and* $s_2 = \beta_0 \cdots \beta_{h-1}$ *holds.*

Since the empty edit operation is explicitly forbidden, an Alignment has a maximum length of $h = |s_1| + |s_2|$. Having defined the alignments it is now possible to formally introduce the Levenshtein distance.

Definition 6 (Levenshtein distance) *The Levenshtein distance of two sequences* s_1 *and* s_2 *is given by*

$$\mathbb{L}(s_1, s_2) = \min \left\{ \sum_{i=0}^{h-1} (\alpha_i \neq \beta_i) \mid \mathcal{A}(s_1, s_2) \right\} \tag{2.5}$$

ie. the minimum number of edit operations $\alpha \rightarrow \beta$ *with* $\alpha \neq \beta$*. The respective alignment is called optimal with respect to the Levenshtein distance.*

Note that there may be more than just one optimal alignment. It is easy to see that $\mathbb{L}(s_1, s_2)$ is upper bounded by $max(|s_1|, |s_2|)$. That would be the case if the characters in s_1 and s_2 were distinct and the differences in length had to be filled with insertions or deletions. To obtain the edit distance of two given sequences in a brute force manner a very large number of aligments needed to be iterated and tested. For long sequences this becomes unfeasible. A way to solve this problem elegantly is a *dynamic programming* algorithm. In brief, the term dynamic programming refers to using the solutions of smaller subproblems to solve a larger problem. Many of the algorithms in computational biology rely on this methodology. First, it needs to be shown that the Levenshtein distance problem can be solved by solving such subproblems. In essence it will formally be proven that, given an optimal alignment of two sequences according to the Levensthein distance, all sub alignments are also optimal.

Theorem 1 *Given an optimal alignment* $\mathcal{A}(s_1, s_2)$ *of* s_1 *and* s_2 *of length* h *such that*

$$\mathcal{A}(s_1, s_2) = \overbrace{(\alpha_1 \rightarrow \beta_1, \cdots, \alpha_{h-2} \rightarrow \beta_{h-2})}^{\mathcal{A}(s'_1, s'_2)} \cdot (\alpha_{h-1} \rightarrow \beta_{h-1}) \tag{2.6}$$

$\mathcal{A}(s'_1, s'_2)$ *is an optimal alignment of* s'_1 *and* s'_2.

The theorem is proven by contradiction.

Proof 1 *Assume* $\mathcal{A}(s_1, s_2)$ *is optimal and* $\mathcal{A}(s'_1, s'_2)$ *is not optimal. Thus*

$\mathcal{A}(s_1', s_2')$ would not deliver the Levenshtein distance of s_1' and s_2'. Hence,

$$\begin{aligned}\mathbb{L}(s_1, s_2) &= \left\{ \sum_{i=1}^{h-2} (\alpha_i \neq \beta_i) \mid \mathcal{A}(s_1', s_2') \right\} + (\alpha_h \neq \beta_h) \\ &> \mathbb{L}(s_1', s_2') + (\alpha_{h-1} \neq \beta_{h-1}) \\ &= \mathbb{L}(s_1, s_2)\end{aligned} \tag{2.7}$$

This is a contradiction. Thus, $\mathcal{A}(s_1', s_2')$ is optimal.

With the formal proof at hand it remains to find the recursions for the dynamic programming algorithm. In principle, at each step it needs to be evaluated whether a substitution, an insertion or deletion minimizes the Levenshtein distance:

$$\mathbb{L}(s_1'\alpha, s_2'\beta) = \min \begin{cases} \mathbb{L}(s_1'\alpha, s_2') + (\varepsilon \neq \beta) & \text{Insertion,} \\ \mathbb{L}(s_1', s_2'\beta) + (\alpha \neq \varepsilon) & \text{Deletion,} \\ \mathbb{L}(s_1', s_2') + (\alpha \neq \beta) & \text{Substitution} \end{cases} \tag{2.8}$$

with $\mathbb{L}(\varepsilon, \varepsilon) = 0$, $\mathbb{L}(\varepsilon, s_2') = |s_2'|$ and $\mathbb{L}(s_1', \varepsilon) = |s_1'|$. Although this equation could directly be implemented using the recursive calls, it is better to tabulate the results in a matrix of size $|s_1| \times |s_2|$. It is easy to see that the dynamic programming algorithm is of $O(|s_1| \cdot |s_2|)$ in time and space. It is important to note that the mere computation of the unit edit distance does not require a full tabulation of all results. In fact it is only necessary to store the results of the previous recursion step to compute the next one. Hence, the Levenshtein distance can be computed with $O(m)$ space requirement.

In several cases, ie. in the comparison of protein sequences, optimal alignments with respect to the unit edit distance are undesirable since some substitutions are more frequent than others. Likewise, some applications require insertions and deletions to recieve higher penalties. The usage of specialized cost functions δ allows such adjustments. A generalized alignment algorithm for the pairwise global alignment using a cost function is the *Needleman-Wunsch algorithm* [Needleman 1970]. It is given by

$$\mathtt{NW}(s_1'\alpha, s_2'\beta) = \max \begin{cases} \mathtt{NW}(s_1'\alpha, s_2') + \delta(\varepsilon, \beta) & \text{Insertion,} \\ \mathtt{NW}(s_1', s_2'\beta) + \delta(\alpha, \varepsilon) & \text{Deletion,} \\ \mathtt{NW}(s_1', s_2') + \delta(\alpha, \beta) & \text{Substitution} \end{cases} \tag{2.9}$$

where typically $\delta(\alpha, \beta) < 0$ for $\alpha \neq \beta$ and $\delta(\varepsilon, \beta) = \delta(\alpha, \varepsilon) < 0$. Similar to the Levenshtein distances we set $\mathtt{NW}(\varepsilon, \varepsilon) = 0$, $\mathtt{NW}(\varepsilon, s_2') = \delta(\varepsilon, \beta) \cdot |s_2'|$

and $\mathtt{NW}(s'_1, \varepsilon) = \delta(\alpha, \varepsilon) \cdot |s'_1|$. In the comparison of protein sequences the cost function δ that specifies the substitution scores often reflects empirical biological observations. The BLOSUM or PAM matrices are two examples of such specialized cost functions. Note, that by multipling the scores by -1 the max operator can be changed to min and vice versa.

In addition to calcuating the similarity of two sequences, the optimal alignment itself is of interest, eg. to find mutations or SNPs in Genomes. Having fully tabulated the results, it is easy to recover the alignments by backtracking the path by simply analyzing which edit operation in the recursion matrix gave rise to the current matrix entry.

2.2 Local alignment

The alignment recursions in the previous section aim to compare two sequences globally. However, in computational biology and other fields it is often necessary to find local similarities of two otherwise distinct sequences. This problem arises, for example, when trying to find similar domains in different proteins or discover similar fragments in two different texts. A dynamic programming algorithm for local sequence comparision has been devised by Smith and Waterman [Smith 1981]. The Smith-Waterman algorithm (SW) aims to find the *optimal local alignment*. Informally, the optimal local alignment recovers the most similar substrings of the two query sequences with respect to some score function δ. Formally,

$$\max \left\{ \sum_{i=1}^{h} \delta(\alpha_i, \beta_i) \mid \mathcal{A}(s'_1, s'_2) \right\} \tag{2.10}$$

where s'_1 and s'_2 are substrings of s_1 and s_2, respectively. Solving this problem by enumeration of all substrings of s_1 and s_2 would require $O(m^3 n^3)$ in time. Fortunately, dynamic programming gives a solution in $O(mn)$. The Smith-Waterman algorithm (SW) differs from the Needleman-Wunsch mainly in its initialization. Leading and trailing insertions are not penalized by the SW algorithm. That is $\mathtt{SW}(\varepsilon, s'_2) = \mathtt{SW}(s'_1, \varepsilon) = \mathtt{SW}(\varepsilon, \varepsilon) = 0$. Furthermore, negative scores are to be suppressed. Thus, the recursion is given by

$$\mathtt{SW}(s'_1\alpha, s'_2\beta) = \max \begin{cases} 0 & \text{Minimum score,} \\ \mathtt{SW}(s'_1\alpha, s'_2) + \delta(\varepsilon, \beta) & \text{Insertion ,} \\ \mathtt{SW}(s'_1, s'_2\beta) + \delta(\alpha, \varepsilon) & \text{Deletion ,} \\ \mathtt{SW}(s'_1, s'_2) + \delta(\alpha, \beta) & \text{Substitution} \end{cases} \tag{2.11}$$

In addition to the global alignment recursions for the Smith-Waterman algorithm it is necessary to keep track of the substrings that recieved the highest score during the recursion. If the results are tabulated, it is easy to obtain the optimal local alignment by back tracking from this maximum entry. It is important to note that for identical sequences the optimal local alignment is identical to the optimal global alignment. We will use this fact later to develop a modified local alignment.

2.3 Semi global alignment

The alignment types introduced above do not fulfill the requirements to optimally align shorter sequences to larger ones. The alignment of DNA-sequencer reads to reference genomes is one example. Since the reference exceeds the query multiple times in length, global alignments are largely irrelevant. Furthermore, it is necessary to align the whole short query, rather than just one of its substrings - rendering also the optimal local sequence alignment useless. Hence, another alignment is necessary. Informally, it is necessary to find an optimal alignment of a substring of the reference s_2' with the whole query s_1. To achieve that a combination of a reference-local and query-global alignment, a *semi-global alignment*, is necessary. To achieve this one can simply modify the initialization of the Needleman-Wunsch algorithm to $\texttt{SG}(\varepsilon, \varepsilon) = 0$, $\texttt{SG}(\varepsilon, s_2') = 0$ and $\texttt{SG}(s_1', \varepsilon) = \delta(\alpha, \varepsilon) \cdot |s_1'|$. This modification allows to shift s_1 along the larger s_2. This is why the semi-global alignment is also referred to as *free shift alignment*. The recursion for the semi-global alignment is the same as for the Needleman-Wunsch alignment. The optimal semi-global alignments can be traced back at the maximum scores in the last row of the matrix. Alternatively, a semi-global alignment with respect to the unit edit distance can be obtained with analogous initialization to the above and the Levenshtein recursions. The trace back is started from the minimum scores in the last matrix row. The algorithm's time complexity is of $O(mn)$. The space requirement is also of $O(mn)$. For large s_2, e.g. in the case of mammalian genomes, this easily becomes unfeasible. If only the positions and the maximum score are of interest, as pointed out above, two columns of length $|s_1|$ suffice and hence the algorithm is $O(m)$ in space. In many applications of semi global alignments a minimum score or maximum unit edit distance is known beforehand. Assume a maximum unit edit distance k has been defined prior to a semi global alignment. Then it suffices to keep $|s_1| + k$ columns of the matrix and the space requirement boils down to $O(m)$. More general, for a minimum score σ_{min} a total of

$$|s_1| + \left\lceil \frac{|s_1| \cdot \arg\max_{\alpha} \delta(\alpha, \alpha) - \sigma_{min}}{\arg\min_{\beta} \delta(\varepsilon, \beta)} \right\rceil \tag{2.12}$$

columns has to be kept to ensure a full trace back at all times.

Proof 2 *Without insertions, an alignment spans at most over $|s_1|$ columns. It remains to show how the alignment length can maximally be increased by the introduction of insertions. The maximum score some sequence s_1 can attain is given by $|s_1| \cdot \arg\max_\alpha \delta(\alpha, \alpha)$. Hence, at most $|s_1| \cdot \arg\max_\alpha \delta(\alpha, \alpha) - \sigma_{min}$ scorepoints can be used up for insertions.*

With the above minimum score the semi global alignment needs $O(m)$ in space.

Myers bit vector algorithm

Semi global alignments are bound by $O(mn)$ in time. However, technical modifications can vastly improve the time consumption. For this purpose Gene Myers has introduced an bit-vector based algorithm that decreases the time for semi global alignments for shorter sequences to large data base sequences from quadratic to linear in practice [Myers 1999]. Informally, this is achieved by a succinct encoding and pre-calculation of alignment columns using a number of bit vectors.

Definition 7 (approximate string matching problem) *The approximate string matching problem is to find all occurences of a sequence in a text with no more than k mismatches, insertions or deletions. Formally, for some query sequence s_1 all positions j in a larger reference sequence s_2 are sought, such that $\mathbb{L}(s_1, s_2[i \cdots j]) \leqslant k$ holds.*

As pointed out above, only the preceeding column of the dynamic programming matrix is required to calculate the current one. As for the Levenshtein distance the neighboring cells of two adjacent columns and rows in the matrix L can at most differ by 1. Formally, for the difference of two horizontal neighbors in the matrix $\Delta h(i,j) = L(i,j) - L(i,j-1)$ and the difference of vertical neighbors $\Delta v(i,j) = L(i,j) - L(i-1,j)$ it was observed that

$$\Delta v(i,j),\ \Delta h(i,j) \in \{-1, 0, 1\}, \quad 0 \leqslant i \leqslant |s_1|,\ 0 \leqslant j \leqslant |s_2| \tag{2.13}$$

by [Ukkonen 1985, Masek 1980] (cf. [Myers 1999]). From this observation it follows for $\Delta x(i,j) = L(i,j) - L(i-j, i-j)$

$$\Delta x(i,j) \in \{0, 1\}, \quad 0 \leqslant i \leqslant |s_1|,\ 0 \leqslant j \leqslant |s_2| \tag{2.14}$$

and hence the diagonal neighbors differ either by 1 or 0. These observations are at the very heart of Myers bit vector algorithm and its extensions. Note that the distance $\mathbb{L}(s_1, s_2[i \cdots j])$ is to be found in the last row of column j. Since the first row in dynamic programming matrices for semi-global alignments is always set to zero, it suffices to know the vector $\Delta v_j =< \Delta v(i,j) >$ of length m to calculate the number of differences with $\sum_i \Delta v_j(i)$ at each j in s_2. Instead of summing up the values of Δv_j there is yet a more efficient way to calculate the distance at the last row using Δh as $L(|s_1|, 0) = |s_1|$ and $L(|s_1|, j) = L(|s_1|, j-1) + \Delta h(m, j)$. To facilitate the computation, let $Peq : \mathbb{N} \times \Sigma \to \{0, 1\}$ be a function with

$$Peq(i, \alpha) = \begin{cases} 1 & \text{iff } s_1[i] = \alpha, \\ 0 & \text{otherwise} \end{cases} \tag{2.15}$$

that can be precomputed and tabulated in an array prior to the alignment. Using the recurrences for the Levenshtein distance we can directly calculate

$$\Delta v(i,j) = (1 - \Delta h(i-1,j)) + \min \begin{cases} -Peq(i, s_2[j]) & \text{Substitution} \\ \Delta v(i, j-1) & \text{Insertion} \\ \Delta h(i-1, j) & \text{Deletion} \end{cases} \tag{2.16}$$

and

$$\Delta h(i,j) = (1 - \Delta v(i,j-1)) + \min \begin{cases} -Peq(i, s_2[j]) & \text{Substitution} \\ \Delta v(i, j-1) & \text{Insertion} \\ \Delta h(i-1, j) & \text{Deletion} \end{cases} \tag{2.17}$$

Since Δv and Δh can be -1, 0 and $+1$, Myers suggests to encode the values by the bit vectors V_j^+, V_j^- and H_j^+, H_j^-, respectively. Where $V_j^+(i) = V_j^-(i) = 0$ iff $\Delta v(i,j) = 0$, $V_j^- = 1 \wedge V_j^+ = 0$ iff $\Delta v(i,j) = -1$ and $V_j^- = 0 \wedge V_j^+ = 1$ iff $\Delta v(i,j) = 1$. The bit vectors for Δh are encoded analogously. With the above equations he gives the following bitwise functions for each entry i of the bit vector j

$$\begin{aligned} H_j^+(i) &= V_{j-1}^-(i) \mid {}^\sim(Peq(i, s_2[j]) \mid H_j^-(i-1) \mid V_{j-1}^+(i)) \\ H_j^-(i) &= V_{j-1}^+(i) \;\&\; (Peq(i, s_2[j]) \mid H_j^-(i-1)) \end{aligned} \tag{2.18}$$

and analogously

$$\begin{aligned} V_j^+(i) &= H_j^-(i-1) \mid {}^\sim(Peq(i, s_2[j]) \mid V_{j-1}^-(i) \mid H_j^+(i-1)) \\ V_j^-(i) &= H_j^+(i-1) \;\&\; (Peq(i, s_2[j]) \mid V_{j-1}^-(i)) \end{aligned} \tag{2.19}$$

while the initialization, strictly along the lines of the semi global alignment, is $H_j^+(0) = H_j^-(0) = 0$. The relevant distance is hence given by

$$L(|s_1|, j) = L(|s_1|, j-1) + (H_j^+(|s_1|)) - (H_j^-(|s_1|)) \tag{2.20}$$

Almost all of the above calculations can in principle be done using bit-parallel operations. Assuming a CPU with word size w and $w > |s_1|$, each vector can be caluclated in a single CPU instruction cycle. The above equations, however, contain a cyclic dependency that inhibits a full bit vector approach. As the calculation of $H^-_j(i)$ requires $H^-_j(i-1)$ and so forth, another formulation needs to be found. The formulation and the solution of this problem makes up a signficant part of Myers ground breaking work. In fact it is shown that

$$H^-_j = (((\mathrm{Peq}(s_2[j]) \;\&\; V^+_{j-1}) + V^+_{j-1}) \wedge V^+_{j-1}) \tag{2.21}$$

and thus H^-_j can also be calculated as a bit vector. Some entry is marked $H^-_j(i) = 1$ if in a preceeding row $k < i$ a match $s_1[k] = s_2[j]$ occurs and all vertical deletion edges $V^+_{j-1}(u) = 1, k < u < i$. Informally, if a match occurs somewhere in an alignment column and the deletion edges $\Delta v(u, j-1)$ strictly increase the distance all the way down to row i, it must follow that $L(i, j-1) > L(i, j)$ and thus $H^-_j(i) = 1$. Having completed all the bit vector formulae Myers arrives at giving his basic bit vector (Alg. 1). Note, that in the algorithm the column subscript indices j are dropped since no trace back is attempted. The space requirement is thus reduced to $O(m)$. For $|s_1| < w$ the time complexity is governed by the precalculaton of the Peq table which is $O(|s_1|\Sigma)$ and the scanning of s_2. Hence, we arrive at an overall complexity of $O(|s_1|\Sigma + |s_2|)$. When $|s_1| \geqslant w$, the complexity is $O(|s_1|\Sigma + |s_2||s_1|/w)$ in time.

Algorithm 1 Myers' bit vector algorithm

```
require $s_1, s_2$
$H^+ = H^- = V^+ i = V^- = 0$
L = 0
for j := 0 to $|s_2| - 1$ do
    $H^+ = V^- \mid \tilde{}(\mathrm{Peq}(s_2[j]) \mid H^- \mid V^+)$
    $H^- = (((\mathrm{Peq}(s_2[j]) \;\&\; V^+) + V^+) \wedge V^+)$
    $L = H^+(|s_1|) - H^-(|s_1|)$
    $H^+ << 1$
    $H^- << 1$
    $V^+ = H^- \mid \tilde{}(\mathrm{Peq}(s_2[j]) \mid V^- \mid H^+)$
    $V^- = H^+ \;\&\; (\mathrm{Peq}(s_2[j]) \mid V^-)$
    if $L \leqslant k$ then
        report match
```

Modifications have been suggested to compute semi-global alignments efficiently also for queries with $|s_1| > w$. Myers gives an extension to integrate Ukkonens block based dynamic programming approach were the column (and thus the query) is split into blocks of size w. The important trick is to skip the calculation of subsequent blocks if the distance calculated in the last row of the current block exceeds k and will exceed k in the same block of the next column. Thus, the basic algorithm is repeated at most $\lceil |s_1|/w \rceil$ times. For larger query sequences a signficant speed up was reported.

Gene Myers did not give an algorithm to recover the alignments from his bit parallel computation. First, to trace back the optimal alignments the basic algorihtm (Alg. 1) has to modified such that a number of bit vector columns have to be stored. With the observation in (Eq. 2.12) it can be seen that the columns for a window of length $s_1 + k$ need to be kept to allow a full trace back at every j that statisfies $\mathbb{L}(s_1, s_2[i \cdots j])$. Heikki Hyyrö has given an algorithm to trace back the alignments using the Δv information and thus only requiring the bit vectors V_j^+ and V_j^- to be stored [Hyyrö 2004]. The intriguing idea is that a vertical move from row i to $i-1$ in column j, i.e. a deletion, in the dynamic programming matrix is only possible if $\Delta v_j(i) = 1$ and thus $V_j^+(i) = 1$. Otherwise, one has to decide if an insertion or substitution occured. To do this the value of $H_j^+(i)$ has to be known. In the following it will be shown that the V^+ and V^- values are sufficient to recover this information. In the following it is assumed that $V_j^+(i) = 0$ and thus $L(i,j) \leqslant L(i-1,j)$.

Theorem 2 *Given* $V_j^+(i) = 0$, *the value of* $H_j^+(i)$ *can be reconstructed from* $V_{j-1}^-(i)$.

With $V_{j-1}^-(i) = 1$ the result $H_j^+(i) = 1$ directly follows from the formulae 2.18. It remains to show that $H_j^+(i) = 0$ if $V_{j-1}^-(i) = 0$. This will be proven by contradiction.

Proof 3 *From* $V_{j-1}^-(i) = 0$ *it follows that* $L(i,j-1) \leqslant L(i-1,j-1)$. *Assume* $H_j^+(i) = 1$ *then* $L(i,j) - 1 = L(i,j-1)$. *We obtain* $L(i,j) - 1 \leqslant L(i-1,j-1)$ *and it follows directly from the observation in Eq. 2.14 that* $L(i,j) = L(i-1,j-1)$ *and hence* $\mathtt{Peq}(i, s_2[j]) = 1$. *This is a contradiction to the formulae 2.18 and so* $H_j^+(i) = 0$.

When $H_j^+(i) = 0$ and $V_j^+(i) = 0$ a substitution edge is on the path of an optimal alignment. Algorithm 2 by [Hyyrö 2004] summarizes the above considerations.

Like other back tracking algorithms, the Hyyroe algorithm is of $O(|s_1| + |s_2|)$ in time. The required space for storing the alignment is linear.

The dynamic programming algorithms, their bit parallel implementations and back tracking approaches are the most important tools in genome informatics and many other fields of computational biology.

2.4 Expectation of a random string

Although the presented improvements can in many cases work in practically linear time, current developments in high throughput biology and

Algorithm 2 A back tracking algorithm for bit vector alignments

```
require matrices $V^+$, $V^-$, start column j
$\mathcal{A} = ()$
$i = |s_1| - 1$
while $i \geqslant 0$ and $j \geqslant 0$ do
   if $V_j^+(i)$ then
      $\mathcal{A} = (s_1[i] \to \varepsilon) \cdot \mathcal{A}$
      i =- 1
   else if $V_{j-1}^-(i)$ then
      $\mathcal{A} = (\varepsilon \to s_2[j]) \cdot \mathcal{A}$
      j =- 1
   else
      $\mathcal{A} = (s_1[i] \to s_2[j]) \cdot \mathcal{A}$
      i =- 1
      j =- 1
$\mathcal{A} = ((s_1[i] \to \varepsilon) \cdots (s_1[0] \to \varepsilon)) \cdot \mathcal{A}$
$\mathcal{A} = ((\varepsilon \to s_2[j]) \cdots (\varepsilon \to s_2[0])) \cdot \mathcal{A}$
return $\mathcal{A}$
```

life science often make their application unfeasible. Heuristic alignment methods have to be found that are able to recover the optimal alignment(s) with good precision and recall rates. Note, that the considerations on the expectation below will only cover perfect matches with the goal to merely sketch some problems that arise from alignment heuristics. The calculation of score based expectation values, such as it is done in BLAST [Altschul 1990] will not be considered here. As briefly introduced in Chapter 1, frequently used heuristics are seed based alignments implemented via hash tables. To measure their effectiveness and efficiency, an important condsideration is the expected number of occurences of some string in a larger reference sequence. Given a probability $p : \Sigma \to [0, 1] \subset \mathbb{R}$ to occur in a reference s_2, the expected number of occurences of a substring s_1 is simpy given by

$$E(s_1) = \Pi_{i=1}^{|s_1|} p(s_1[i]) \cdot |s_2| \tag{2.22}$$

Assuming that queries of length k and reference sequences of length n are uniformly and randomly composed this boils down to $E = p^k \cdot n$. Thus, we arrive at

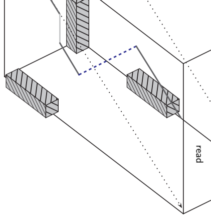
$$k = \log_p(E/n) \tag{2.23}$$

to directly calculate the length k depending on the expectation and n. Setting the expectation to $E = 1$ we obtain

$$k = \log_p(1/n) = -\log(n)/\log(p) \tag{2.24}$$

As an example, with $p = 0.25$ and large reference sequences of the size of mammalian genomes with more than 3 billion nucleotides, e.g., $k \approx 16$. In other words in a (uniformly random) mammalian genome a (uniformly random) query is expected to occur more than once if the query length is shorter than 16 nucleotides. Since real life genomes (and real life queries) are not uniformly randomly distributed the above considerations are crude at best. In fact, to amplify non-reptitive regions in a genome, primer sequences with more than 20 nucleotides are needed to ensure specificity.

Seed based heuristics that aim to perfectly match a part of the query s_1 with the larger reference s_2 have to balance sensitivity and specificity. If the seeds are to short, the number of matches in the reference quickly becomes unfeasible and renders the approach useless. Furthermore, if perfect matching seeds are required, potential errors will either result in a wrong seed alignment (especially if a short seed is chosen) or not produce any seed alignment at all (if a long seed is chosen). Circumventing these pitfalls while at the same time achieving acceptable computation times is an important goal of this thesis.



Chapter 3

Suffix arrays

Contents

3.1 The suffix trie and tree 29
  3.1.1 Linear time construction of suffix trees 32
3.2 Suffix arrays 33
  3.2.1 Linear time construction of suffix arrays 35
3.3 Enhanced suffix arrays 38
  3.3.1 lcp-table 39
  3.3.2 child table 40
  3.3.3 suffix link intervals 45

As pointed out above not only current genome sequencing projects are producing enourmous amounts of data. Likewise vastly growing internet based text corpora pose a challenge to hard- and software recources. Indexing is one important strategy to quickly search those data resources. Two techniques to index texts are suffix trees and suffix arrays. For better understanding of the the latter data structure, suffix trees are briefly described. The algorithms for the construction of suffix arrays and enhanced suffix arrays described here have been implemented in a C software library in the context of [Hoffmann 2009].

3.1 The suffix trie and tree

The suffix trie is a data structure that is able to implicitly represent all substrings of a given text. Informally, the suffix trie is a rooted tree with edges labeled by characters. For a sequence S of length n there are n leafs and thus n distinct paths form the root down to a leaf. Each leaf represents one suffix text. An example of the suffix trie for the string $abab\$$ is given in Fig. 3.1. In the following the notation $S_i = S[i \cdots n]$ to denote the i^{th} suffix of S will be used and i is the *suffix number*. The *leaf number* is the suffix number which the leaf represents.

Note, that a special character, the sentinel $\$ \notin \Sigma$, has been added to the string $abab$. Without the sentinel, the above informal description would not hold: the suffix ab is nested inside $abab$ and there is no leaf for ab in the suffix tree for $abab$ only (cf. Fig. 3.1). To enforce a leaf for ab, a

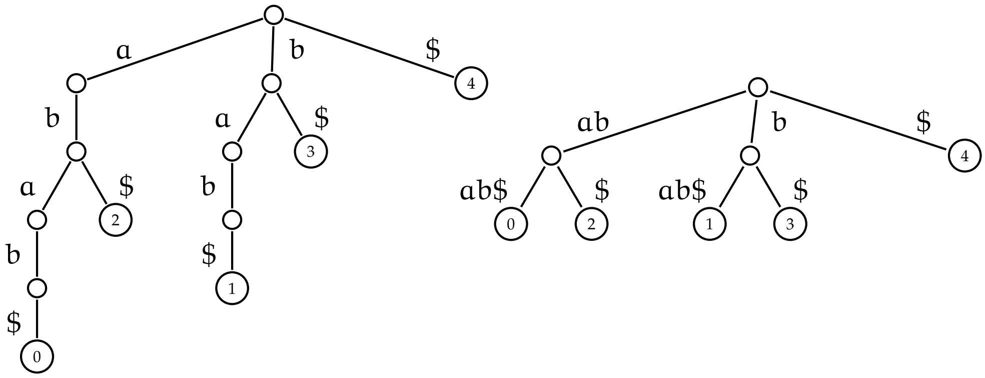


Figure 3.1: A suffix trie (left) and the corresponding suffix tree (right) for the sequence $abab\$$. In contrast to the trie, the suffix tree has no internal nodes with only one successor. Moreover, the edges in the trie are labeled by characters, in the tree by non-empty strings. The leafs of the tree are labeled by the suffix number.

sentinel character that does only occur at the end of the sequence needs to be added.

Suffix tries are very similar to suffix trees. The only formal difference in trees is their *compactness*. While in suffix tries edges are only labeled by single characters from the alphabet Σ, the edges in suffix trees are labeled by strings over Σ^+. The compactness is achieved by removing all nodes that only have one outgoing edge from the suffix trie. Both data structures are equivalent. We arrive at

Definition 8 (suffix trie) *A suffix trie* $\mathsf{STE}(S\$) \subset (E, V)$ *for a sequence* $S \in \Sigma^+$ *is a graph where*

1. *edges are labeled by characters* $\alpha \in \Sigma$
2. *only one edge label per node begins with same* $\alpha \in \Sigma$
3. *the concatenated edge labels of a path through* $\mathsf{STE}(S\$)$ *are substrings of S\$.*
4. *represents all substrings of* $S\$$

From the above definition it follows that for each substring $v \in S$ there is an internal vertex that represents v and vice versa. In the following v will be used to identify an internal node. In contrast to the trie, the suffix tree is compact and not every substring is represented by a node. Hence, the suffix tree has advantages in the implementation since a smaller number of internal nodes needs to be explicitly represented. For this reason we will only use suffix trees in the remainder of this manuscript.

Definition 9 (suffix tree) *The definitions 8.2 to 8.4 for* $\mathsf{STE}(S\$)$ *also hold for the suffix tree* ST. *For a suffix tree for a sequence* $S \in \Sigma^+$ *it is additionally required that*

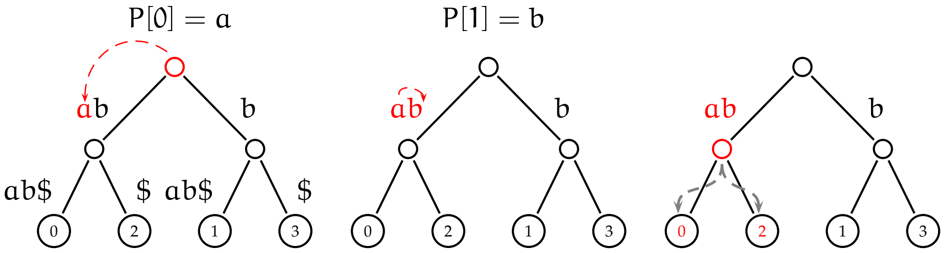


Figure 3.2: Matching a query $P = ab$ in the suffix tree $ST(S\$)$ for $S = abab$. The matching path ends in the node w for the sustring ab. The leafs of the subtree of w direclty indicate the matching positions of P in S

1. *edges are labled by strings* $w \in \Sigma^+$
2. *compact, ie. has no internal nodes with less than two successors*

Having defined the suffix tree formally, some of its features that will be of importance can be investigated later. Note, that each vertex has at most $|\Sigma| + 1$ children. Searching a query sequence P of length m in a suffix tree is easy. Starting with the root and the first character of P, at each node an edge label av is sought such that $P[i] = a$. Since $|\Sigma|$ is constant, this can be done in constant amortized time. If such an edge exists, the search is continued within the label. For each match i is incremented. The search terminates if $i = m$ or a mismatch occurs. In case $i = m$ the positions of P in S can directly be looked up in the suffix numbers of the leafs of the corresponding subtree.

Another property is the capability to quickly find all *minimum unique substrings* of a sequence S.

Definition 10 (minimum unique substring) *A minumum unique substring is a substring* va *that occurs only once in* S *while* v *occurs more than once.*

In the example $abab$ there are only two minimum unique substrings, aba and bab. To obtain these substrings from the compact suffix tree we only need to enumerate the internal nodes of $ST(S\$)$ that are connected to leafs l. For all edges that lead to a leaf l with $v \xrightarrow{aw} l$ with edge label $aw \in \Sigma^+$, va is a unique minimum substring. With the considerations in the above chapter the length of va can be estimated for a uniformly random sequence by $k = \log_p(1/n) = -\log(n)/\log(p)$ where $p = 1/|\Sigma|$. In other words after matching k characters we can expect to be on an edge towards a leaf in a suffix tree. Thus, for query sequences P with $P = S[i \cdots j]$ of length $m >> k$ we only need to match approximately k characters to retrieve the position i. A similar situation can be observed in answering the existence question in a suffix tree. Assuming a query $P \not\subseteq S$, we expect to be able to match approximately k characters along the edge labels before a first mismatch occurs and the exitstence questions can be answered negatively. To answer the existence question positively, however, it is necessary to fully match P. Nevertheless, the above considerations show that some questions can be answered in sublinear time and it becomes apparent that the suffix tree is a powerful data structure. Along the same lines, with $ST(S\#T\$)$ of two sequences S and T that are concatenated by a special symbol $\# \notin \Sigma$ it is possible to answer additional questions such as finding maximal repeats, i.e. the longest substrings that occur in S and T, in linear time.

3.1.1 Linear time construction of suffix trees

An important question is how to construct suffix trees. To efficiently do this also for large sequences S, a linear time construction algorithm is desirable. A number of algorithms have been devised so far (cf. [McCreight 1976, Ukkonen 1995, Giegerich 1997]. In the following a brief sketch of the linear time construction algorithm devised by McCreight [McCreight 1976] is given. The idea is to build the suffix tree for $S\$$ by successively adding the suffixes form $S\$_0$ to $S\$_n$. It exploits the concept of the longest common prefix.

Definition 11 (longest common prefix) *The longest common prefix of two strings* S *and* T *of length* m *and* n*, respectively, is a sequence* $S[0 \ldots i] = T[0 \ldots j]$ *with* $i \in [0 \ldots m], j \in [0 \ldots n]$ *such that* $S[0 \ldots i+1] \neq T[0 \ldots j+1]$.

McCreight uses the fact that a suffix that is to be added to the current tree may be partioned it two parts using the longest common prefix.

Definition 12 (head and tail) *The* $\mathsf{head}(i)$ *of suffix* $S\$_i$ *is the longest common prefix* $S\$[i \ldots i+l-1]$ *with a suffix* $S\$_j, j < i$*. The* $\mathsf{tail}(i)$ *denotes the rest* $S\$[l+1 \ldots n]$*. Hence* $S\$_i = \mathsf{head}(i)\mathsf{tail}(i)$.

Once the suffix tree for the suffixes $S\$_0..S\$_i$ is iteratively build, only parts of the suffix tree that are affected by introducing the new suffix $S\$_{i+1}$ need to be updated. Assuming a longest common prefix with $S\$_{i+1}$ it will be proven that all substrings of $head(i+1)$ are already properly represented. Only $\mathsf{tail}(i+1)$ needs to be inserted at a proper position. Formally,

Theorem 3 *All substrings of* $head(i+1)$ *are already represented in the current suffix tree build from* $S\$_0..S\$_i$ *.*

Let $lcp : \Sigma^+ \times \Sigma^+ \to \mathbb{N}$ be a function that returns the length of the longest common prefix denoted by $lcp(S,T) = \ell$.

Proof 4 *Let* $\mathsf{ST}(S\$)$ *be a suffix tree of* S, $a \in \Sigma$, $v \in \Sigma^+$. *Assume a string* av *of length* m *that has a longest common prefix* $\ell > 1$ *with the suffixes represented in* $\mathsf{ST}(S\$)$. *Hence, there is a path through* $\mathsf{ST}(S\$)$ *of length* $\mathrm{lcp}(av, S) = \ell$. *It directly follows for* v *that* $\mathrm{lcp}(v, S) \geqslant \ell - 1 \geqslant 1$ *and* $v[0 \cdots \ell - 2]$ *is also a substring of* S *and thus represented in* $\mathsf{ST}(S\$)$.

Hence, we only need to add $\mathsf{tail}(i+1)$ below the location of $\mathsf{head}(i+1)$ in current tree. $\mathsf{head}(i+1)$ implicitly points to the edge or vertex where $\mathsf{tail}(i+1)$ has to be added. Finding $head(i+1)$ and its location $\mathsf{loc}(\mathsf{head}+1)$, ie. the tree node or the tree edge that represents $\mathsf{head}(i+1)$ in the tree, could be done by searching the tree. From the above proof it follows that for $\mathsf{head}(i) = \ell > 1 \Rightarrow \mathsf{head}(i+1) \geqslant 1$ and it is not necessary to rescan the whole suffix $S\$_{i+1}$. In fact, the location of $\mathsf{head}(i)$ that we already know from the previous iteration can be used to find $\mathsf{head}(i+1)$ much quicker. This is realized using suffix links.

Definition 13 (suffix link) *Let* $av, a \in \Sigma, v \in \Sigma^*$. *If the location* $\mathsf{loc}(av)$ *in* $\mathsf{ST}\$$ *is an internal node, the suffix link is defined as* $\mathsf{suflink}(\mathsf{loc}(av)) = \mathrm{loc}(v)$. *If* $v = \varepsilon$, $\mathsf{suflink}(\mathsf{loc}(av))$ *points to the root.*

Note that the suffix links are only defined for nodes but not edges. To analyse how suffix links are used in McCreight's algorithm assume we want to insert the new suffix $S\$_{i+1} = S\$[i+1 \cdots n]$ and know the location $\mathsf{loc}(\mathsf{head}(i))$ for $\mathsf{head}(i) = avw$. For $\mathsf{loc}(avw)$ we dont have a suffix link yet. However, as the substring v was already represented by the suffix tree before the insertion of $S\$[i \cdots n]$ we have a suffix link for the parent node $\mathsf{suflink}(\mathsf{loc}(av)) = \mathsf{loc}(v)$. After taking the link it remains to rescan w to arrive at $\mathsf{loc}(vw)$. Now, the suffix link can be updated with $\mathsf{suflink}(\mathsf{loc}(avw)) = \mathsf{loc}(vw)$ and we continue scanning the new suffix for $\mathsf{head}(i+1)$. Finally, $\mathsf{tail}(i+1)$ is added.

To complete the algorithm 3 let $\mathsf{search} : (V, E) \times \Sigma^* \rightarrow V \times \Sigma^*$ be a function that starts a search for $w \in \Sigma^*$ at location $p \in (V, E)$. It returns the matched prefix of w and the location where the search stopped. The function $\mathsf{parent} : (V, E) \rightarrow V \times \Sigma^*$ returns the parent node of a location and the edge label between the location and the parent. Note, that the search function used here is a simplification. In fact, after the suffix link is taken, it is possible to use a fast skip-and-count function that does not require character comparisions. The algorithm runs in O(n). It was shown that, although using quite different concepts, McCreights algorithm similar to Ukkonens online construction algorithm [Giegerich 1997].

3.2 Suffix arrays

First we introduce basic notions for the suffix array. In contrast to the suffix tree, a suffix array is a much simpler data structure. Informally, the

Algorithm 3 McCreight's M algorithm for suffix tree construction

```
require $ST(S\$[0..n]) \subset (V, E)$, text $S\$$
for $i$ in $0$ to $n$ do
  if $\mathsf{head}(i) = \varepsilon$ then
    $(\mathsf{loc}(\mathsf{head}(i+1)), \mathsf{head}(i+1)) = \mathsf{search}(\varepsilon, S\$[i+1..n])$
    if $\mathsf{loc}(\mathsf{head}(i+1)) \in E$ then
      add a node at $\mathsf{loc}(\mathsf{head}(i+1))$
    attach leaf with number $i+1$ to $\mathsf{loc}(\mathsf{head}(i+1))$ with label $\mathsf{tail}(i+1)$
  else
    $(p, w) = \mathsf{parent}(\mathsf{loc}(head(i)))$
    $p = \mathsf{suflink}(p)$
    $(\mathsf{loc}(p), p) = \mathsf{search}(p, w)$
    if $\mathsf{loc}(p) \in E$ then
      $\mathsf{head}(i+1) = p$
      $\mathsf{loc}(\mathsf{head}(i+1)) = \mathsf{loc}(p)$
    else if $\mathsf{loc}(p) \in V$ then
      $(\mathsf{loc}(\mathsf{head}(i+1)), \mathsf{head}(i+1)) = \mathsf{search}(p, \mathsf{tail}(i))$
    if $\mathsf{head}(i+1) \in E$ then
      add a node at $\mathsf{loc}(\mathsf{head}(i+1))$
    attach leaf with number $i+1$ to $\mathsf{loc}(head(i+1)$ with label $\mathsf{tail}(i+1)$
    $\mathsf{suflink}(\mathsf{loc}(\mathsf{head}(i))) = \mathsf{loc}(p)$
```

suffix array is a sorted list of all suffixes S_i of some S. While searching a substring in a suffix array can not be done in linear time as in the suffix tree, it already allows the application of binary search algorithms and thus reduces the complexity to $O(m \log n)$. Due to its simplicity it uses significantly less space compared to the suffix tree. The data structure has been simultaneously developed by Gonnet et al. [Gonnet 1992] and Gene Myers and Udi Manber [Manber 1993] independently pusblished the structure under the name suffix array. In their work they present a first enhancement ot the suffix array, the lcp-table lcp that helps to further reduce the complexity of searches. The lcp-table lcp will be covered in the next section.

Like in the case of the suffix tree, a sentinel character \$ (not occurring in S) is added to the end of S. The concept of suffix arrays is based on lexicographically sorting the suffixes of S\$. Suppose that the characters are ordered such that, i.e. for a DNA alphabet $\mathtt{A} < \mathtt{C} < \mathtt{G} < \mathtt{T} < \mathtt{N} < \$$. This character order induces an order on all non-empty suffixes of S\$, which is captured in the suffix array. Formally,

Definition 14 (suffix array) *the suffix array* suf *of* S *is an array of integers in the range* 0 *to* n*, specifying the lexicographic order of the* $n+1$ *non-empty suffixes of* $S\$$*. In other words,* $S_{\mathsf{suf}[\$]}0$, $S_{\mathsf{suf}[\$]}1$, …, $S_{\mathsf{suf}[\$]}n$ *is the sequence of suffixes of* S *in ascending lexicographic order.*

Searching for all occurences of string P of length m in the suffix array means finding the first $S_{\mathsf{suf}[\$]}u$ and the last $S_{\mathsf{suf}[\$]}v$ such that $S_{\mathsf{suf}[\$]}u[0 \cdots m-1] = S_{\mathsf{suf}[\$]}v[0 \cdots m-1] = P$. For searching a sequence P in S, Manbers and Myers give a simple binary search algorithm to find u (Alg. 4). v is sought in the same way. Note, that at the end of (Alg. 4) it needs to be checked if $S_{\mathsf{suf}[\$]}u[0 \cdots m-1] = P$. Let $occ_S(P)$ denote the set of occurrences, i.e. the set of positions i, $0 \leqslant i \leqslant |S| - m$ satisfying $P = S[i..i + m - 1]$. $occ_S(P)$ can be found by applying Alg. 4.

Algorithm 4 Binary search algorithm for string searches in suffix arrays

```
require $P$, $S$, suf
if $P \leqslant S_{\mathsf{suf}[\$]}0[0 \cdots m-1]$ then
  return $0$
else if $P > S_{\mathsf{suf}[\$]}n-1[0 \cdots m-1]$ then
  return $n$
else
  $L = 0$
  $R = 0$
  while L - R > 1 do
    $M = (L + R)/2$
    if $P \leqslant S_{\mathsf{suf}[\$]}M[0 \cdots m-1]$ then
      $R = M$
    else
      $L = M$
  return $L$
```

3.2.1 Linear time construction of suffix arrays

Since the introduction of suffix arrays it has been known that suffix arrays can be constructed from suffix trees in linear time. Due to the fact that suffix trees themselves can be constructed in linear time, e.g. by McCreights algorithm, it was clear that the overall construction time of suffix arrays was of $O(n)$. A direct linear time construction algorithm, however, was missing for several years. In [Manber 1993] a divide-and-conquer algorithm that proceeds by partioning a set of suffixes by their first character is described. This divide-phase has $\log(N + 1)$ stages. In total we arrive at a time complexity of $O(n \log n)$. More than ten years later Ko and Aluru presented a linear time construction algorithm for suffix arrays [Ko 2005]. Several other algorithms have been proposed [Baeten 2003, Kärkkäinen 2003, Schürmann 2007]. In this work the linear suffix array construction algorithm of Ko and Aluru was used and is sketched in the following. The key observation of their linear time construction algorihtm is that suffixes can be partioned in a natural fashion.

Definition 15 (K and G suffixes) *A suffix $S\$_i$ is of type K if $S\$_i < S\$_{i+1}$ and of type G otherwise. The last suffix has no successor and is therefore of both types.*

All suffixes of $S\$$ can be classified K or G in a linear $O(n)$ scan. The classification is easy if $S\$_i[0] \neq S_i[1] = S\$_{i+1}[0]$. If $S\$_i[0] = S\$_{i+1}[0]$ the scan just needs to be continued to the first position k where $S\$_i[0] \neq S\$_k[0]$. If $S\$_i[0] < S\$_k[0]$, all suffixes between are of type K and of type G otherwise. The following theorem is key to Ko and Alurus suffix sorting algorithm.

Theorem 4 *Type K suffixes are lexicographically greater than type G suffixes if they start with the same character.*

In contrast to [Ko 2005] the proof is given constructively.

Proof 5 *Assume two suffixes of type $S\$_i = auv$ and $S\$_j = auw$ of type K and type G, respectively. It directly follows directly from the definition that $auv < v$ and $auw > w$. Where u is possibly empty. Since $v[0] > w[0]$, auv is lecicographically greater than auw.*

This theorem is a significant advantage, as we know that the type K suffixes appear after the type G suffixes in the lexicographically sorted suffix array. Assume we have found a way to sort all K suffixes. It remains to sort the type G suffixes. This can be done in linear time and the following observation helps to do it. The smallest suffix $S\$_n = \$$ is by definition also of type K and thus it appears first in the lexicographically ascending list of K-suffixes. Naturally, $S\$_{n-1} = a\$$ must be the lexicographically smallest suffix that begins with character a. If we had bucketed the suffixes according to their first characters, $S\$_{n-1}$ needed to be pushed to the front of the a-bucket and the front needed to be increased by one. Increasing of the front is necessary because we already know that $S\$_{i-1}$ must be the smallest suffix in its bucket. This is the trick that can be used to sort all type G suffixes in linear time.

Given the above consideration, assume, we have bucketed all suffixes according to their first characters and the type K suffixes are already sorted. Obviously, we have at most $|\Sigma| + 1$ buckets and the buckets can be ordered without extra effort according to the lexicographical order. The (lexicographically greater) K suffixes appear at the right end of each bucket and are already sorted.

Theorem 5 *If we iterate over all buckets in lexicographical order, and scan each bucket from left to right. For each suffix $S\$_i$ that is encountered, $S\$_{i-1}$ is located and pushed to the front of its respective bucket if it is of type G. After each push the front is increased by one. In this way all type G suffixes can be sorted.*

Since we are iterating the buckets in lexicographical order and scan them strictly form left to right, it remains to proof that some suffix S_i that is encountered during the scan is already correctly sorted in its bucket.

Proof 6 *Since all type* K *suffixes are already sorted the proof can be restricted to type* G *suffixes. If* $S\$_i = a u \alpha$ *is of type* G *there is a suffix* $S\$_j = \alpha, i < j$ *such that* $S\$_i > S\$_j$*. Since* $S\$_j$ *is lexicographically smaller than* $S\$_i$*, it has been moved to the front of its bucket in a previous step of the scan. If there is a* $S_k = a u \beta$ *with* $\beta < \alpha$ *it has been moved to the front of the bucket before* S_i *was moved to the front and since the front was increased* S_k *resides left of* S_i*. Likewise, suffixes with* $\beta > \alpha$ *reside right of* S_i*. Thus,* S_i *is already sorted using the instruction form the above theorem.*

Algorithm 5 summarzies the linear sorting of G-suffixes. A reverse array R keeps track of the position of each suffix $S\$_i$ in the buckets B, such that $B[S\$[i]][R[i]] = i$. The algorithm begins by writing all suffixes $S\$_i = a\nu$ to their respective buckets representing a. Next, the list of sorted K-suffixes is used to push the K-suffixes to right end of their respective buckets.

Algorithm 5 Aluru sorting algorithm

```
require sorted list $L$ of K-suffixes
init arrays reverse $R$, bucket $B$, front array $F$
for $i := 0$ to $n$ do
  $R[i] = |B[S\$[i]]| + 1$
  add $i$ to bucket $B[S\$[i]]$
for $i := |L| - 1$ to $0$ do
  $\kappa = L[i];\ s = S\$[\kappa];\ l = |B[s]|$
  swap $B[s][R[\kappa]]$ with $B[s][l - F[s]]$
  update $R$
  $F[s] += 1$
clean and initialize $F$
for $a \in \Sigma$ do
  for $i := 0$ to $|B[a]|$ do
    $\pi = B[a][i] - 1$
    if $S\$[\pi]$ is type G suffix then
      swap $B[S\$[\pi]][R[\pi]], B[S\$[\pi]][F[s]]$
      update $R$
      $F[s] + = 1$
return B
```

Since there are n suffixes in the buckets algorithm 5 runs in $O(n)$ and has a space complexity of $O(n)$.

Now it remains to find a way to sort type K suffixes. The idea is to iteratively sort the K-substrings by repeated re-bucketing.

Definition 16 (K-substrings) *K-substrings are the substrings between two type K suffixes.*

At the end each K-substring is represented by a bucket B and the buckets are sorted in lexicographical sorted by the lexicographical order. The bucket holds all occurences of the K-substring in $S\$$.

The procedure has three steps. First, for all suffixes of $S\$$ the distance to the preceding K suffix is calculated. That is, the distance of $S\$_i$ to the previous K-suffix is stored in $D[i]$. In the same iteration, all K-suffixes are bucketed according to their first character. In the next stage, the information stored in $D[i]$ is used to construct lists L such that $L[d][a], d \in \mathbb{N}, a \in \Sigma$ holds all suffixes with distance d to the previous K-suffix and $S\$[\pi] = a, \pi \in L[d][a]$. Informally, in the last step the K-substrings are iteratively sorted according to their second character, third character and so forth using the the lists L. Therefore, we iterate over the d from smallest to greatest and over all characters in lexicographical order. For each $\pi \in L[d][a]$, K-suffix $\pi - i$ is moved to the front of a new bucket $C[a]$. The re-bucketing to $C[a]$ ensures that only equal substrings of length i remain in the same bucket. At the end of each iteration B is replaced by C. Finally, we have calculated the bucket list B that holds all substrings in lexicographical order (Alg. 6). Again, the array R keeps track of the positions of the suffixes in the buckets B.

Note that each K-suffix starts with a prefix of a K-substring. Thus, we can replace all K-substrings in $S\$$ with the corresponding bucket number in the sorted list. The resulting string S' is a sequence of integers. In other words, by bucketing, sorting and labeling the K-substrings are encoded as characters of an integer alphabet. The numerical order of the integer alphabet reflects the lexicographical order of the K-substrings. The resulting string S' can in turn be classified in K and G suffixes and recursively sorted. The recursive algorithm terminates, when all characters of the alphabet occur only once in S'. Since there are at most $\lceil \frac{n}{2} \rceil$ suffixes in S' the complete algorithm runs in $O(n)$ in space and time.

3.3 Enhanced suffix arrays

The enhanced suffix array (ESA) is the combination of the suffix array, the lcp-table, the child-table and the suffix link interval table. The ESA was introduced by [Abouelhoda 2004]. While suffix arrays are not equivalent to the suffix trees, ESA are [Abouelhoda 2004]. Just like in the suffix tree, exact searches of strings in the ESA require only linear time. The improvement comes at the cost of an increased space requirement. In the following the additional tables will be informally and formally introduced.

Algorithm 6 K-substring sorting algorithm

```
$k = 0$
for $i := 0$ to $n-1$ do
  $D[i] = k$
  if $S[i]$ is K-suffix then
    $k = 1$
    add $i$ to $B[S\$[i]]$
    update $R$
  else if $k > 0$ then
    $k = k + 1$
for $i := 0$ to $n-1$ do
  if $D[i] > 0$ then
    add $i$ to list $L[D[i]][S\$[i]]$
for $i := 1$ to $\max(D)$ do
  initialize $F$
  for $a \in \Sigma$ do
    for $j := 1$ to $|L[i][a]|$ do
      $\pi = L[i][a][j] - i;\ s = S\$[\pi]$
      $C[a][F[a]] = B[s][R[\pi]]$
      update $R$
      $F[a] + = 1$
  $B = C$
return B
```

3.3.1 lcp-table

The lcp-table is a table that holds the length of the longest common prefix of two subsequent suffixes of S in lexicographical order.

Definition 17 (lcp-table) *The lcp-table* lcp *is an array of integers in the range* 0 *to* $n-1$. *For each* h, $1 \leqslant h \leqslant n$, $\mathsf{lcp}[h]$ *is the length of the longest common prefix of* $S_{\mathsf{suf}[h-1]}$ *and* $S_{\mathsf{suf}[h]}$.

Since the suffix $S\$_n = \$$ is the last suffix in the lexicographic order of all non-empty suffixes, $S_{\mathsf{suf}[\$]}n = \$$. Hence we always have $\mathsf{lcp}[n] = 0$.

The lcp-table can also be constructed in linear time. Once the longest common prefix h for some suffix i is determined, it directly follows from the considerations for McCreights algorithm that the longest common prefix of the suffix $i+1$ is at least of length $h-1$. Hence, for $S\$_{i+1}$ the character comparision of the first $h-1$ characters can be skipped. If a reverse suffix table such that $\mathsf{rev}[\mathsf{suf}[i]] = i$ is available, the longest common prefix of S_i with S can be determined by directly jumping into the suffix table (Alg. 7).

From algorithm 7 it can directly be seen that each character of $S_i[h]$ is compared only once. Hence, the algorithm runs in $O(n)$. The lcp-table is usefull to construct another enhancement - the child table.

Algorithm 7 Linear time construction of the the lcp table

```
$h = 1$
for $i := 0$ to $n$ do
  $j = \mathsf{rev}[i]$
  if $j > 0$ then
    $h- = 1$
    $k = \mathsf{suf}[j-1]$
    while $S\$_i[h] = S\$_k[h]$ do
      $h+ = 1$
    $\mathsf{lcp}[j] = l$
```

3.3.2 child table

Runs of some minimum value $\mathsf{lcp}[j] \geqslant h > 0, i \leqslant j \leqslant k$ indicate that the sorted suffixes $S_{\mathsf{suf}[j]}$ have a common prefix of at least h. Given a suffix array for $S \in \Sigma^+$ there are at most $|\Sigma| + 1$ entries with $\mathsf{lcp}[j] = 0, 0 \leqslant j \leqslant n$. These values partition the table in *lcp intervals*, one lcp interval for each character that occurs in S. These intervals correspond to the children of the root of the equivalent suffix tree. Using this simple observation, the whole suffix tree can be reconstructed from the lcp table. The idea was introduced in [Abouelhoda 2004].

Definition 18 (lcp-intervals) *An interval* $[l..r, h]$ *is a lcp interval if the following holds:*

1. $0 \leqslant l \leqslant r \leqslant n$
2. $0 \leqslant h \leqslant n + 1$
3. $\mathsf{lcp}[i] \geqslant h$ *for all* $i, l + 1 \leqslant i \leqslant r$
4. $l = 0$ *or* $\mathsf{lcp}[l] < h$
5. $r = n$ *or* $\mathsf{lcp}[r + 1] < h$
6. $\mathsf{lcp}[i] = h$ *for at least one* $i, l + 1 \leqslant i \leqslant r$

For some h, the lcp-interval is called h-interval. A h-interval $[l..r, h]$ refers to table suf, denoting the set $\varphi([l..r, h]) = \{\mathsf{suf}[j] \mid l \leqslant j \leqslant r\}$ of suffixes of S$ such that all suffixes of S$ in $[l..r, h]$ have a common prefix, say w, of length h. Vice versa, all suffixes of S$ having prefix w are in $[l..r, h]$. Due to this correspondence, we say that $[l..r, h]$ *is the lcp interval for* w. $r - l + 1$ is the *width* of $[l..r, h]$.

The construction of the lcp-intervals can be done in linear time. The algorithm introduced in [Kasai 2001] was modified by [Abouelhoda 2004]. It uses a simple stack. While scanning the suffix array from top to bottom lcp-values $\mathsf{lcp}[i]$ that are greater than the current value on the top of the stack are pushed, together with position $i-1$, onto the stack. If an lcp value $\mathsf{lcp}[r]$ smaller than the lcp value at the top, say h, is encountered, $[i-1\cdots r-1, h]$ can be reported as a h-interval. This procedure simulates a bottom-up traversal of the *lcp interval tree* of S$ in linear time and reports all h-intervals of the array in bottom-up order. The lcp-interval tree is basically a suffix tree without leaves. Hence, an lcp interval is equivalent to an internal node of the suffix tree of S$.

For string matching purposes, however, a top-down traversal of the interval tree would be desirable. Fortunately, the sketched algorithm can be used to construct a *child table* that allows the retrieval of all children of a given lcp-interval (internal node of the suffix tree) in constant time and thus allows the desired traversal.

To do this, a few more observations are necessary. Assume we have a h-interval $[l, r]$. Analogously to the example at the beginning of this subsection there are at most $|\Sigma|$ entries with $\mathsf{lcp}[k] = h, l+1 \leqslant k \leqslant r$. We refer to these entries as h-indices.

Theorem 6 (children of a h-interval) *The h-indices of a h-interval $[l, r]$ in ascending order, namely l_1, l_k, indicate the starts of nested lcp-intervals $[l, l_1-1][l_1, l_2-1]\cdots[l_k, r]$ that are the children of the interval $[l, r]$.*

It needs to be shown that all the above intervals are children of $[l, r]$.

Proof 7 *By definition, all suffixes in $[l, r]$ share the first h characters. For all h-indices we have*

$$S_{\mathsf{suf}[l_j]}[h+1] \neq S_{\mathsf{suf}[l_j-1]}[h+1] \tag{3.1}$$

and since l_j differs in the next possible character with l_{j-1} it must be the first suffix of a child interval of $[l, r]$. Since there are no other intervals possible by construction the h-indices define all the children of $[l, r]$.

It now remains to find all h-indices in a given lcp-interval. To do that Abouelhoda et al. propose three additional tables, up, down and $\mathsf{next}h$, and later show a trick to reduce the space requirement. Keep in mind that whenever we see an $\mathsf{lcp}[i-1] > \mathsf{lcp}[i]$ an lcp-interval is ending. Informally, for each lcp-interval $[l, r]$, up or down point to the first h-index h_1 and for some h-index h_j, $\mathsf{next}h$ points to the next h-index. Formally,

$$\begin{aligned} \mathsf{up}[i] &= \min\{j \mid j \leqslant i-1, \mathsf{lcp}[j] > \mathsf{lcp}[i], \\ &\qquad \mathsf{lcp}[k] \geqslant \mathsf{lcp}[j],\ j+1 \leqslant k \leqslant i-1\} \\ \mathsf{down}[i] &= \max\{j \mid j \geqslant i+1, \mathsf{lcp}[j] > \mathsf{lcp}[i], \\ &\qquad \mathsf{lcp}[k] > \mathsf{lcp}[j],\ i+1 \leqslant k \leqslant j-1\} \end{aligned} \tag{3.2}$$

Note, that finding the first h-index in $[l, r]$ is equivalent to using a range minimum query [Fischer 2007].

$$\mathsf{down}[l] = \mathsf{RMQ}_{\mathsf{lcp}}(l+1, r) = \min\{k \mid \mathsf{lcp}[k] \leqslant \mathsf{lcp}[q],\ k, q \in [l+1, r]\} \quad (3.3)$$

For some interval $[l, r]$, $\mathsf{up}[r+1]$ and $\mathsf{down}[l]$ point to the first h-index of an enclosed interval. This redundancy will later be resolved. It remains to compute the $\mathsf{next}h$ table.

$$\begin{aligned} \mathsf{next}h[i] = \min\{j \mid j \geqslant i{+}1, \mathsf{lcp}[j] = \mathsf{lcp}[i], \\ \mathsf{lcp}[k] \geqslant \mathsf{lcp}[i],\ i{+}1 \leqslant k \leqslant j{-}1\} \end{aligned} \quad (3.4)$$

Hence, at some h-index $\mathsf{next}h$ points to the next h-index and thus subsequent child interval. With the sketch of a bottom-up traversal of the suffix interval tree the two additional tables can be constructred in linear time using (Alg. 8) by Abouelhoda et al. A thorough proof of correctness is given there [Abouelhoda 2004]. As described above it uses a stack S and the typical stack operations push, pop and top.

Algorithm 8 Construction of up and down table

```
last = −1
push(S, 0)
for i := 1 to n do
    while $\mathsf{lcp}[i] < \mathsf{lcp}[\mathsf{top}(S)]$ do
        last = pop(S)
        if $\mathsf{lcp}[i] \leqslant \mathsf{lcp}[\mathsf{top}(S)] \wedge \mathsf{lcp}[\mathsf{top}(S)] \neq \mathsf{lcp}[last]$ then
            down[top(S)] = last
    if $last \neq -1$ then
        up[i] = last
        last = −1
    push(S, i)
```

It now remains to construct the $\mathsf{next}h$ table. From the considerations above it is clear that we only need to find all $\mathsf{lcp}[k] = h$. The algorithm (Alg. 9) was introduced in [Abouelhoda 2004]. This table can also be constructed in linear time.

Algorithm 9 Construction of the $\mathsf{next}h$ table

```
init stack $S = \perp$
push(S, 0)
while $\mathsf{lcp}[i] < \mathsf{lcp}[\mathsf{top}(S)]$ do
    pop(S)
if $\mathsf{lcp}[i] = \mathsf{lcp}[\mathsf{top}(S)]$ then
    nexth[pop(S)] = i
push(S, i)
```

The information in up and down is redundant.In fact the three tables can be merged to a single table to reduce the space requirements down to a third. To achieve that we need to make some important observations. Note that the $\mathsf{up}[j_k + 1] = \ell_k$ stores the first h-index of the lcp-interval $[i_k, j_k]$ enclosed in $[l, r]$. Furthermore, the h-index to the next interval $[i_{k+1}, j_{k+1}]$ is stored in $\mathsf{nexth}[i_k] = i_{k+1}$. For the last child $[i_q, r]$ of the interval $[l, r]$ however the $\mathsf{up}[r + 1]$ is already outside of the current interval. But since $\mathsf{nexth}[i_q]$ is empty (its the last interval) the $\mathsf{down}[i_q]$ can be stored in $\mathsf{nexth}[i_q]$. Thus we have eliminated the down table without deleting information. From the definition of the h-index we already know how to check wheter a value is a next h-index or a down value pointing to the first ℓ_q.

Theorem 7 *A value in the merged table* $\mathsf{nexth}[i]$ *for next* h*-index can be distinguished from a* down *value by checking if* $\mathsf{lcp}[\mathsf{nexth}[i]] = \mathsf{lcp}[i]$

The proof follows directly.

Proof 8 *All* h*-indices in* $[l, r]$ *have the same lcp-value. Thus the condition evaluates true if the value is the next* h*-value. Since the value of* $\mathsf{down}[i_q]$ *points to the first* ℓ*-index inside the interval* $[i_q, r]$ *it must follow* $\mathsf{down}[i_q] > \mathsf{nexth}[i_q]$ *and thus the claim.*

There is yet one more space optimization. At the end of each interval $[l, r]$ the $\mathsf{nexth}[r]$ is necessarily empty because we have reached the end of the interval and $[r, r]$ cannot have any more children. Thus we are able to store the information in up in the nexth table too. To test if the value is from up, it is simply necessary to test for the end of the interval by $\mathsf{lcp}[i] > \mathsf{lcp}[i + 1]$. Now we have eliminated the second table and have constructively shown that only one table, now called child table, is necessary to store the information for a top-down traversal of the suffix interval tree. To get the first h-index we introduce algorithm 10. Note, that to test if the first h-index of $[l, r]$ is stored in $\mathsf{child}[r]$ it suffices to check $l < \mathsf{child}[r] \leqslant r$. In summary the space requirement for the child table is $O(n)$ integers, i.e. 4n bytes.

Algorithm 10 get first h index for interval $[l, r]$ from child table

```
require interval $[l, r]$
if $l < \mathsf{child}[r] \leqslant r$ then
    $i1 = \mathsf{child}[r]$
else
    $i1 = \mathsf{child}[l]$
return i1
```

Finally, we are able to enumerate all children of a given interval (Alg. 11).

Algorithm 11 Enumerating all child intervals of $[l, r]$

```
require $[l, r]$
**if** $l > 0 \lor r < n - 1$ **then**
  $i1 = \mathsf{getfirsthindex}(l, r)$
  report child $[i, i1 - 1]$
**else**
  $i1 = i;$
**while** $i1 < n \land \mathsf{lcp}[\mathsf{child}[i1]] = \mathsf{lcp}[i1] \land \mathsf{lcp}[i] > \mathsf{lcp}[i + 1]$ **do**
  $i2 = \mathsf{child}[i1]$
  report child $[i1, i2 - 1]$
  $i1 = i2$
report child $[i1, r]$
```

The algorithm reports all children of a given interval. Alternatively, the algorithm may return a set M of all children. Let $\mathsf{getchild} : \mathbb{N}^2 \times \Sigma \to \mathbb{N}^2$ be a function that returns a child interval $[u, v] = \mathsf{getchild}(u, v, a)$ such that $S[\mathsf{suf}[u]] = S[\mathsf{suf}[v]] = a$. Using Algorithm 11 with a constant alphabet size $|\Sigma|$ this can be done in constant armotized time. Now pattern matches can be done in linear time in the enhanced suffix array. The algorithm 12 has been sketched by [Abouelhoda 2004].

Algorithm 12 Exact pattern matches in enhanced suffix arrays

```
require pattern $P$ of length $m$
$found = 1$
$i = 0$
$[l, r] = \mathsf{getchild}(0, n, P[i])$
**while** $[l, r]$ exists and $i < m$ and $found = 1$ **do**
  **if** $l \neq r$ **then**
    $h = \mathsf{lcp}[\mathsf{getfirsthindex}(l, r)]$
    $j = \min(h, m) - 1$
    $found = (S[\mathsf{suf}[l] + i \cdots \mathsf{suf}[l] + j] = P[i \cdots j])$
    $i = j + 1$
    $[l, r] = \mathsf{getchild}(0, n, P[i])$
  **else**
    $found = (S[\mathsf{suf}[l] + i \cdots \mathsf{suf}[l] + m - 1] = P[i \cdots m{-}1])$
**if** found **then**
  report matches in $[l, r]$
```

Finding the interval $[l, r]$ such that all suffixes start with the same character takes at most $|\Sigma| + 1$ comparisions and thus can be done in constant amortized time for a fixed alphabet size. The above algorihtm performs exactly m character comparisions and thus runs in $O(m)$. Later it was shown that the above description of the child table can be more naturally explained using *Super-Cartesian trees* [Fischer 2010]For the algorithm for short read mapping introduced in this manuscript yet one more enhancement is needed.

3.3.3 suffix link intervals

In their publication on enhanced suffix arrays Abouelhoda et al. devised an algorithm to also reconstruct the suffix link information of the equivalent suffix tree in $O(n \log n)$. Maaß [Maaß 2007] later devised a linear time algorithm. To understand the idea, the description of the linear suffix tree construction of McCreight (Alg. 3) comes in handy. Whenever a new node is created on an existing edge, we know that $\mathsf{suflink}(\mathsf{loc}(\mathsf{head}(i))) = \mathsf{loc}(\mathsf{head}(i+1))$. Because the construction in McCreights algorithm proceeds in a linear fashion from the longest to the shortest suffix, it follows for all internal nodes $p \in ST$, that the second smallest leaf number, say ℓ, indicates the suffix S_ℓ that p was created for. Furthermore, it follows that the suffix link of the internal node created for $S_{\ell-1}$ points to p. Vice versa, for some internal node $u \in ST$ with the second smallest suffix number ℓ we have to find the $p \in ST$ that has been created for $S_{\ell+1}$ and thus has the second smallest leaf number $\ell+1$. Thus, the suffix links can be reconstructed by a bottom-up traversal to store all internal vertices p as a function of their second smallest leaf numbers followed by a top-down traversal to establish the links. The algorithms (Alg. 13, 14) are a modification of the algorithm introduced by Maaß to work on suffix arrays. During the bottom-up traversal (Alg. 13), the relation of the sought second smallest suffix number $\ell+1$ and the $\mathsf{child}[r]$ is stored in an array A such that $A[\ell+1] = r$. The bottom-up traversal of the suffix array is similar to (Alg. 8). It is realized using the stack S_1. The second stack S_2 holds the left borders of the child intervals along with the minimum suffix number (leaf number) for each interval. During the subsequent top-down traversal (Alg. 14) we store the depth of h-intervals in array B, such that $B[h]$ holds the last seen h-interval. Since we perform a top-down traversal, this is the ancestor of depth h for each leaf encountered. For the top-down traversal we can directly use the child table information.

Given the above considerations we know that the interval $[l, r]$ represented by r needs to be linked to an internal node with the second smallest interval $\ell+1$.

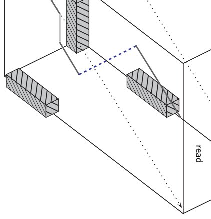
In the implementation used here a less space consuming approach is used. Note that most of the intervals $[l, r]$ are small and often $r - l < 255$. Thus it suffices to store only one position, say l, and calculate r using a one byte id tab such that $l + \mathsf{id}(l) = r$. In cases where $l - r \geqslant 255$ we can store the width in a seperate sorted array that can be accessed via a binary search. Thus the space requirement for the $\mathsf{suflink}$ table is reduced from $8n$ for two integer values to $5n$ bytes. The same idea also decreases the space requirement for the lcp table.

Algorithm 13 bottom-up traversal to create suffix link map A

```
push($S_1$, $0$)
push($S_2$, $(0, \mathsf{suf}[0])$)
for $i := 1$ to $n$ do
    $lb = i-1$
    push($S_2$, $(lb, \mathsf{suf}[lb])$)
    while $\mathsf{lcp}[i] < \mathsf{lcp}[\mathsf{top}(S_1)]$ do
        $llb = \mathsf{pop}(S_1)$
        /*child interval is $[llb, i-1]$*/
        min1 = min2 = n+1;
        while $llb \leqslant \mathsf{top}(S_2)[0] \leqslant i-1$ do
            $(lb, m) = \mathsf{pop}(S_2)$
            if $m < min1$ then
                $min2 = min1$
                $min1 = m$
            else if $m < min2$ then
                $min2 = m$
        $lb = llb$
        push($S_2$, $(lb, min1)$)
        $A[min2 + 1] = [lb, i-1]$
    if $\mathsf{lcp}[i] > \mathsf{lcp}[\mathsf{top}(S_1)]$ then
        push($S_1$, $lb$)
```

Algorithm 14 top-down traversal to establish suffix links

```
push($S$, $[0, n]$)
while $S$ is not empty do
    $[a, b] = \mathsf{pop}(S)$
    $h = \mathsf{lcp}[\mathsf{getfirsthindex}(a, b)]$
    $B[h] = [a, b]$
    for all child intervals $[l, r]$ of $[a, b]$ do
        if $l < r$ then
            push($S$, $[l, r]$)
        else
            $[u, v] = A[\mathsf{suf}[l]]$
            if $u > 0 \wedge v < n-1$ then
                $i1 = \mathsf{getfirsthindex}(u, v)$
                $h = \mathsf{lcp}[i1]$
                $\mathsf{suflink}[i1] = B[h-1]$
```



Chapter 4

A heuristic method for short read alignment

Contents

4.1 Outline of a novel mapping approach 48
4.2 Methods 52
4.3 Results 63

Current evaluations indicate that high-throughput sequencing methods produce reads with very different error rates and error types (see Introduction). While the most frequent error-type in Illumina reads are mismatches, reads produced by 454's GS FLX predominantly contain insertions and deletions (indels). The long reads of Pacific Biosystems Sequencers have substantially higher error rates especially at the 3'-end. Fast and accurate matching for these diverse reads in is therefore still a practical problem. In addition, up until today the handling of multiple matches varies substantially between the different mapping tools [Fonseca 2012]. We have introduced a matching model that is able to cope with mismatches and indels, addresses different error models and works for reads of arbitrary length. For example, it can handle the problem of leading and trailing contaminations caused by primers and poly-A tails in transcriptomics or the length dependent increase of error rates. In these contexts it thus simplifies the tedious and error-prone trimming step. For efficient searches, our method utilizes index structures in the form of enhanced suffix arrays. In a comparison with current methods for short read mapping, the approach presented here shows significantly increased recall not only for 454 reads but also for Illumina reads. Our approach is implemented in the software `segemehl` [Hoffmann 2009] available at `http://www.bioinf.uni-leipzig.de/Software/segemehl/`. In a recent survey, the benchmarks of `segemehl`compare favourably to other state of the art mappers [Otto 2014]. Independent evaluations of our tool are shown in the Discussion of this Thesis.

4.1 Outline of a novel mapping approach

The development of read mapping methods decisively depends on specifications and error models of the respective sequencing technologies. Unfortunately, little is known about specific error models, and the models are likely to change as manufactures are constantly modifying chemistry and machinery. Increasing the read length is a key aim of all vendors — tolerating a trade-off with read accuracy. Investigation on the typical error models of 454 and Illumina technologies showed that 454 reads are more likely to include insertions and deletions while Illumina reads typically contain mismatches [Huse 2007, Dohm 2008]. In fact, indels account for more than two thirds of the errors found in 454 reads. However, the first generation of read mapping programs did not consider indels in the seeding step, e.g. `MAQ` [Li 2008a], `ZOOM` [Lin 2008a], `SOAP` [Li 2008b], `SHRiMP`, `Bowtie` [Langmead 2009], and `ELAND` (proprietary).

In addtion, some of these programs, in particular `SOAP` and `MAQ`, were specifically designed to map short Illumina or SOLiD reads and did not accept longer reads. For this reason, several of these algorithms have been extended, e.g. Bowtie2 [Langmead 2012a], BWA-SW and BWA-MEM [Li 2013].

The manufacturers' overestimations of read accuracies often spoil optimistic expectations of the success of the read mapping. While an overall error rate of 0.5% has been observed for 454, the error rate increases drastically for reads shorter than 80 bp and longer than 100 bp [Huse 2007], leading to considerably larger error frequencies in real-life datasets. This implies that sequencing projects aiming to find short transcripts such as miRNAs lose a substantial fraction of their data, unless a matching strategy is used that takes indels into account. In Illumina reads, error rates of up to 4% have been observed [Dohm 2008]. This differs significantly from Illumina's specification. Compared to 454, the frequency of indels is significantly lower. Moreover, differences between reads and reference genome might also occur due to genomic variations such as SNPs. Newer sequencing technologies such as Pacific Biosystems produce even longer reads and come, according to the information of the manufacturer, with an error rate of 10%.

Here, we present a matching method that uses enhanced suffix arrays to compute exact and inexact seeds to account for all of the above error types and is able to map reads of arbitrary length. Sufficiently good seeds subsequently trigger a full dynamic programming alignment. Our method is insensitive to errors and contaminations at the ends of a read including $3'$ and $5'$ primers and tags. The results section describes the basic ideas and an evaluation of our `segemehl` software implementing our method. The technical details of the matching model are described in the Methods section at the end of this contribution.

A read aligner should deliver the original position of the read in the reference genome. Such a position will be called the *true position* in the following. Optimally scoring local alignments of the read and the reference genome can be used to obtain a possible true position, but because an alignment of the read with the reference genome at the true position does not always have an optimal score according to the chosen scoring scheme, this method does not always work. Nevertheless, there are no better approaches available unless further information about the read is at hand.

We present a new read mapping approach that aims at finding optimally scoring local alignments of a read and the reference genome. It is based on computing inexact seeds of variable length and allows to handle insertions, deletions (indels; gaps), and mismatches. Throughout this thesis the notion of differences refers to mismatches, insertions and deletions in some local alignment of the read and the reference genome, irrespective of whether they arise from technical artifacts or sequence variation. A single difference is either a single mismatch, a single character insertion or a single character deletion. Although not limited to a specific scoring scheme, we have implemented our seed search model in the program `segemehl` assigning a score of 1 to each match and a score of -1 to each mismatch, insertion or deletion. Our matching strategy derives from a simple and commonly used idea. Assume an optimally scoring local alignment of a read with the reference genome with exactly two differences. If the positions of the differences in the alignment are sufficiently far apart, we can efficiently locate exact seeds which in turn may deliver the position of the optimal local alignment in the reference genome. Likewise, if the distance between the two differences is small, two continuous exact matches at the ends of the read possibly allow to map the read to this position. To exploit this observation, the method presented employs a heuristic based on searches starting at all positions of the read. That is, for each suffix of the read the longest prefix match, i.e. the longest exact match beginning at the first position of the suffix with all substrings of the reference genome is computed. If the longest prefix match is long enough that it only occurs in a few positions of the reference genome, it may be feasible to check all these positions to verify whether the longest prefix match is part of a sufficiently good alignment. While this approach works already well for many cases, we need to increase the sensitivity for cases where the computation of the longest prefix match fails to deliver a match at the position of the optimally scoring local alignment. This is the case when a longer prefix match can be obtained at another position of the reference genome by exactly matching characters that would result in a mismatch, insertion or deletion in the optimal local alignment (cf. Fig. 4.1).

Therefore, during the computation of each longest prefix match we

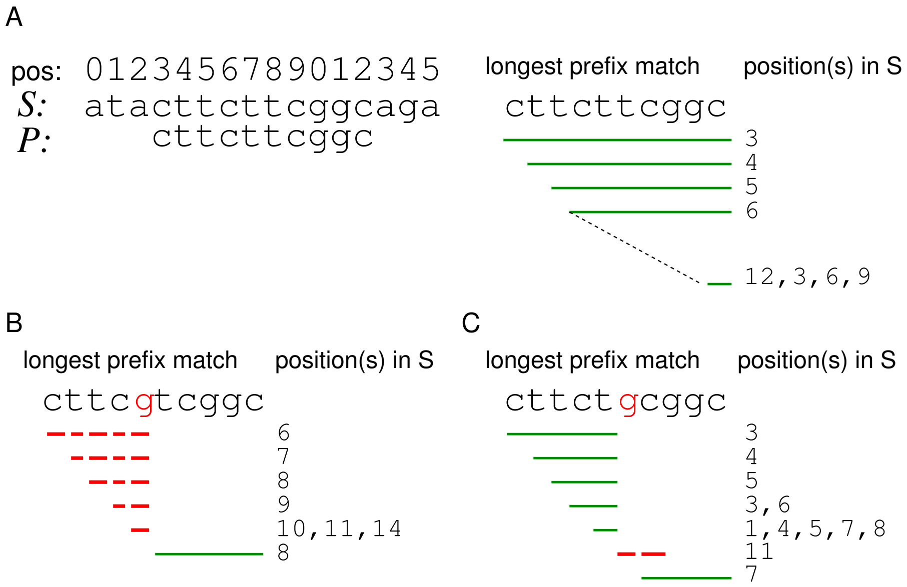


Figure 4.1: **Longest prefix matches may fail to deliver the position of the optimally scoring local alignment.** Assume a simple scoring scheme that assigns a score of $+1$ to a single character match and a score of 0 to a single character mismatch, a single insertions or deletion. Using longest prefix matches bears the risk of ignoring differences in the best, i.e. optimally scoring, local alignment. Its retrieval fails if a longer match can be obtained at another position of the reference sequence by matching a character, that is inserted, deleted, or mismatched in the best local alignment. Depending on the length of the reference genome and its nucleotide composition the probability is determined by the length of the substring that can be matched to the position of the best local alignment before the first difference occurs. (A) The optimally scoring alignment of the read $P :=$ `cttcttcggc` begins at position 3 of the reference genome $S :=$ `atacttcttcggcaga`. Let P_i denote the i^{th} suffix of the read P. For each P_i, the starting positions of the longest match in S comprise the position of P_i in the best local alignment (solid green lines). That is, the longest match of P_0 begins at position 3, the longest match of P_1 begins at position 4, the longest match of P_2 begins at position 5 and so forth. (B) For the read $P :=$ `cttcgtcggc`, the retrieval of the best local alignment fails for all P_i, $i < 5$ (dashed red line) due to the inclusion of a character that results in a mismatch in the optimally scoring local alignment. (C) The read $P :=$ `cttctgcggc` contains, with respect to the best local alignment, a mismatch at position 5 of the read. Here the position 5 of the read is not included in the longest prefix match and nearly all substrings align correctly to the reference genome.

check a limited number of differences by enumerating all possible mismatches and indels (at certain positions) cf. Fig. 4.2).

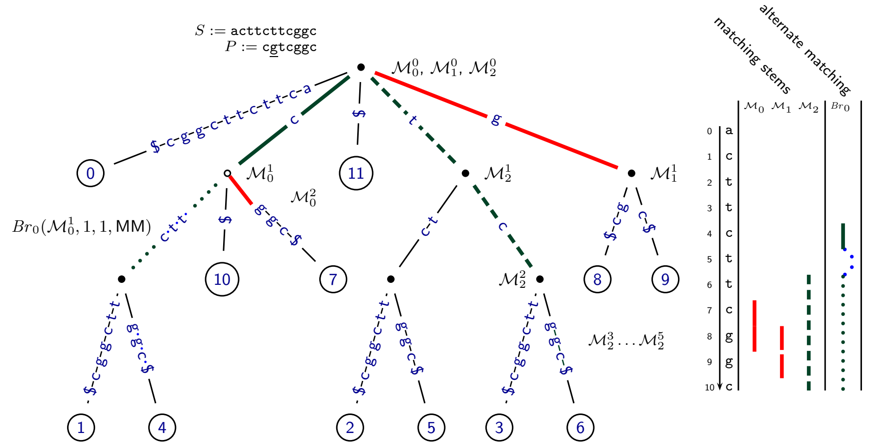


Figure 4.2: **Matching stems and matching branches.** We give an explanation based on a suffix trie which is equivalent to the suffix interval tree shown in Fig. 4.3 (see Methods). The suffix trie for S$ with S := `acttcttcggc` (left) holds twelve leaves. Each numbered leaf corresponds to exactly one suffix in S. Nodes with only one child are not explicitly shown. Note, that internal nodes implicitly represent all leafs in their respective subtree. Thus, internal nodes can be regarded as sets of suffixes. The right panel holds the longest matches for different matching paths in the trie. Matching the first three suffixes of the read P := `cgtcggc` results in three different paths in the suffix trie. Each path is equivalent to a sequence of suffix intervals, a matching stem, in the enhanced suffix array. Let $\mathcal{M}_i$ denote the matching stem for $P_i = i^{th}$ suffix of P. The q^{th} interval in $\mathcal{M}_i$, denoted by $\mathcal{M}_i^q$, implicitly represents the set of suffixes in S matching $P[i..i+q-1]$. The path for the first suffix P_0 is of length two (green solid line). Hence, the equivalent matching stem $\mathcal{M}_0$ is a sequence of three intervals: $\mathcal{M}_0^0$, $\mathcal{M}_0^1$ and $\mathcal{M}_0^2$. Since $\mathcal{M}_0^2$ only represents the suffix S_7, the longest prefix match of P_0 is of length 2 occurring at position 7 of the reference sequence (right panel). The matching stem $\mathcal{M}_1$ for P_1 (red solid line) ends with $\mathcal{M}_1^2$. Therefore, matches of length one occur at positions 8 and 9 in S. The longest prefix match for P_3 occurs at position 6 of S (dashed orange line). Note, that the intervals $\mathcal{M}_3^3 \cdots \mathcal{M}_3^5$ of $\mathcal{M}_3$ equivalently represent S_6. An alternative path leads to a match with position 4. The branch $Br_0(\mathcal{M}_0^1, 1, 1, \text{MM})$ denotes the alternative that accepts the mismatch of g and t at position 1 of P_0.

To efficiently compute the longest prefix matches, we exploit their properties for two consecutive suffixes of a read, i.e. for two suffixes starting at position i and i + 1. If the suffix starting at position i has a longest prefix match of length ℓ, then the suffix starting at position i + 1 has a longest prefix match of length at least ℓ − 1. For example, assume a read `ACTGACTG`. If the second suffix has a longest prefix match of length 4, i.e. `CTGA`, with the reference genome, we immediately see that the third suffix has a longest prefix match not shorter than 3—because we already know that the substring `TGA` exists in the reference genome. Using an enhanced suffix array of the reference sequence, we can easily exploit this fact and determine the longest prefix match of the next suffix without rematching the first $\ell - 1$ characters. Likewise, the enumeration of mismatches and indels is also restricted to the remaining characters of the suffix in our model.

For each suffix of a read, we thus obtain a set of exact matches and alternative inexact matches and their respective positions in the reference sequence. These exact and inexact matches act as seeds. A seed is omitted if it occurs more than t times in the reference genome, where t is a user specified parameter (`segemehl` option `-maxocc`). The heuristics rigorously selects the exact or inexact seed with the smallest E-value, computed according to the Blast-statistics [Karlin 1990]. If this E-value is smaller than some user defined threshold (`segemehl` option `-E`), the bitvector algorithm of [Myers 1999] is applied to a region around the genomic position of the seed to obtain an alignment of the read and the reference sequence. While the score based search for local alignment seeds controls the sensitivity of our matching model, the bitvector alignment controls its specificity: if the alignment has more matching characters than some user specified percentage α of the read (`segemehl` option `-A`) the corresponding genomic position is reported (see Methods).

The computation of the longest prefix match is implemented by a top-down traversal of a conceptual suffix interval tree, guided by the characters of the read. The suffix interval tree is equivalent to a suffix trie (see Methods). The traversal delivers a matching stem. Note that for the DNA alphabet there are at most four edges outgoing from each node of the suffix interval tree. To introduce mismatches, the traversal is simply continued with alternative edges, i.e. edges diverging from the matching stem. To introduce insertions, the traversal is not regularly continued, but characters of the read are skipped. Deletions are simulated by skipping nodes of the suffix interval tree and continuing the search at their child nodes (see Methods). We refer to these alternative paths that branch off from the matching stem as branches. The maximum number of branches to be considered is controlled by the seed differences threshold k (`segemehl` option `-D`). Note, that while matching character by character along a suffix of a read, the number of branches is expected to decrease quickly.

4.2 Methods

Our strategy, based on enhanced suffix arrays, aims to find a best local alignment of short reads and reference sequences with respect to a simple scoring system. It does so by determining, for each suffix of the read, the longest prefix occurring as a substring in the reference sequence. This gives a matching backbone, from which a limited number of branches is derived by mismatches, insertions and deletions (Fig. 4.2). The concept of matching backbone is equivalent to the concept of matching statistics introduced in [Chang 1994]. We introduce the concept of matching backbone and branches via a conceptual tree of suffix intervals. Our heuristic approach delivers a small number of inexact seeds of variable length that are subsequently checked by the bitvector algorithm of Myers [Myers 1999] to verify the existence of alignments with a limited number of differences. First, a short introduction to the basic notions for sequence processing and enhanced suffix arrays will be given, before the concept of suffix intervals is defined. Subsequently, we introduce our new matching strategy.

Basic notions for sequence processing

We consider sequences over the DNA alphabet $\Sigma_{\mathrm{DNA}} = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}, \mathtt{N}\}$, where `N` denotes an undetermined base. In our approach the alignment of `N` with any character, including `N` itself, results in a mismatch.

Matching concept

We now formally introduce the notion of *suffix intervals* that is very similar to the lcp interval and at the heart of the matching strategy in enhanced suffix arrays presented below.

Definition 19 (suffix interval) *An interval* $[l..r, h]$ *is a suffix interval if the following holds:*

1. $0 \leqslant l \leqslant r \leqslant n$
2. $0 \leqslant h \leqslant n+1$
3. $\mathsf{lcp}[i] \geqslant h$ *for all* $i, l+1 \leqslant i \leqslant r$
4. $l = 0$ *or* $\mathsf{lcp}[l] < h$
5. $r = n$ *or* $\mathsf{lcp}[r+1] < h$

The notion of suffix intervals slightly generalizes the notion of lcp-intervals. A suffix interval $[l..r, h]$ of width at least 2 is an *lcp-interval* if, besides condition 1.-5. above, we additionally have $\mathsf{lcp}[i] = h$ for at least one i, $l+1 \leqslant i \leqslant r$. This condition requires that at least one pair of consecutive suffixes in the suffix interval has a longest common prefix of length exactly h (Fig. 4.3). In other words, a suffix interval $[l..r, h]$ of width 2 which is not an lcp-interval does not have a maximum lcp-value h, implying that $[l..r, h+1]$ is also a suffix interval.

i	suf	lcp	$S_{\mathsf{suf}[i]}$
0	0	0	*acttcttcggc*\$
1	7	0	*cggc*\$
2	4	1	*cttcggc*\$
3	1	4	*cttcttcggc*\$
4	10	1	*c*\$
5	9	0	*gc*\$
6	8	1	*ggc*\$
7	6	0	*tcggc*\$
8	3	2	*tcttcggc*\$
9	5	1	*ttcggc*\$
10	2	3	*ttcttcggc*\$
11	11	0	\$

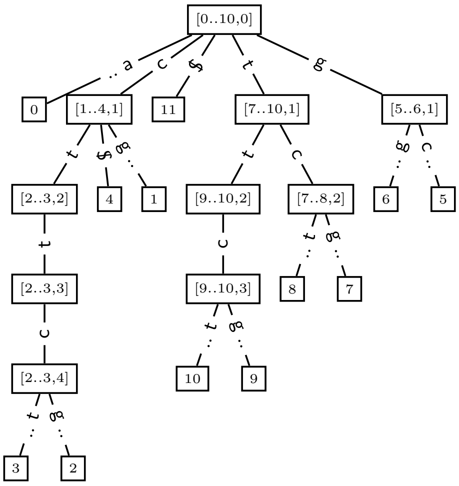


Figure 4.3: **The enhanced suffix array yields a tree structure of nested suffix intervals.** The enhanced suffix array for the sequence $S :=$ `attcttcggc` (left) and its suffix interval tree (right), equivalent to the suffix trie in Fig. 4.2, is shown. The array suf represents the lexicographical order of the suffixes in S\$. In other words, $S_{\mathsf{suf}[0]}, S_{\mathsf{suf}[1]}, \ldots, S_{\mathsf{suf}[n]}$ is the sequence of suffixes of S\$ in ascending lexicographic order. The lcp-table lcp is an array of integers such that for each h, $1 \leqslant h \leqslant n$, $\mathsf{lcp}[h]$ is the length of the longest common prefix of $S_{\mathsf{suf}[h-1]}$ and $S_{\mathsf{suf}[h]}$. A suffix interval $[l..r, h]$ denotes an interval in the suffix array with $\mathsf{lcp}[i] \geqslant h$ for all $i, l+1 \leqslant i \leqslant r$, i.e. all suffixes in the interval $[l+1..r]$ have a longest common prefix of length at least h. Additionally, requiring $l = 0$ or $\mathsf{lcp}[l] < h$ makes the suffix interval left maximal and requiring $r = n$ or $\mathsf{lcp}[r+1] < h$ makes it right maximal. The suffix interval $[0..10, 0]$ spans the whole suffix array and is equivalent to the root of a suffix interval tree. This interval contains five subintervals, one for each character in S\$, with $h = 1$. Equivalently, the root node of the suffix interval tree has five children. Note, that two children, labeled by 0 and 11, are singletons. The child nodes of singletons are not explicitly shown here.

While suffix intervals correspond one-to-one to the nodes of a suffix trie for S\$ (cf. [Crochemore 2007]), lcp-intervals correspond to the branching nodes of a suffix tree for S\$ (cf. [Abouelhoda 2004]). Interpreting the additional condition for lcp-intervals for trees means that in suffix trees nodes with a single child are omitted, while they are allowed in suffix tries. Note that $\varphi([l..r, h]) = occ_S(w)$ whenever $[l..r, h]$ is the suffix interval for w. Consider the suffix interval $[l..r, h]$ for w. A child of $[l..r, h]$ is a suffix interval $[l'..r', h+1]$ satisfying $l \leqslant l' \leqslant r' \leqslant r$. We call $[l'..r', h+1]$ the a-child of $[l..r, h]$ if there is a character $a \in \Sigma_{\mathrm{DNA}}$ such that $[l'..r', h+1]$ is the suffix interval for wa. Note that for all q, $l' \leqslant q \leqslant r'$, we have $a = S_{\mathsf{suf}[q]}[h]$. Hence we can easily determine a from $[l'..r', h+1]$ or split $[l..r, h]$ into its children. A method computing the a-child of a suffix interval in constant time is described in [Abouelhoda 2004].

Let $\mathcal{M} = [l..r, h]$ be a suffix interval. For the empty sequence ε we define

$\mathsf{E}(\mathcal{M}, \varepsilon) = \mathcal{M}$. For any character a and any sequence u we recursively define

$$\mathsf{E}(\mathcal{M}, au) = \begin{cases} \mathcal{M} & \text{if there is no } a\text{-child of } \mathcal{M} \\ \mathsf{E}(\mathcal{M}', u) & \text{otherwise} \end{cases}$$
$$\text{where } \mathcal{M}' \text{ is the } a\text{-child of } \mathcal{M}$$

That is, $\mathsf{E}(\mathcal{M}, v)$ delivers the interval $\mathcal{M}'' = [l''..r'', q]$, obtained by greedily matching the characters in v beginning at the suffix interval $\mathcal{M}$ and q is the length of the matching prefix of v.

Let P denote a sequence of length m neither containing a wildcard symbol $\mathtt{N}$ nor the sentinel \$. For any $i, 0 \leqslant i \leqslant m-1$, $P_i = P[i..m-1]$ denotes the suffix of P beginning at position i. Let ℓ_i be the length of the longest prefix of P_i occurring as a substring of S. Then $P[i..i+\ell_i-1]$ occurs in S and either $i+\ell_i = m$ or $P[i..i+\ell_i]$ does not occur in S. Moreover, there is a sequence of suffix intervals $\mathcal{M}_i = (\mathcal{M}_i^0, \mathcal{M}_i^1, \ldots, \mathcal{M}_i^{\ell_i})$, such that for all q, $0 \leqslant q \leqslant \ell_i$, $\mathcal{M}_i^q$ is the suffix interval for $P[i..i+q-1]$. This implies that $\varphi(\mathcal{M}_i^q) = \mathit{occ}_S(P[i..i+q-1])$ (see chapter on suffix arrays). We call $\mathcal{M}_i$ a matching stem. Obviously, for any $i, 0 \leqslant i \leqslant m$, $\mathcal{M}_i^0 = [0..n, 0]$. For any $i, 0 \leqslant i \leqslant m$ and any q, $1 \leqslant q \leqslant \ell_i$, $\mathcal{M}_i^q$ is the a-child of $\mathcal{M}_i^{q-1}$ where $a = S[t+q-1]$ for any $t \in \varphi(\mathcal{M}_i^q)$. (Note that all suffixes in $\mathcal{M}_i^q$ have the common prefix $P[i..i+q-1]$ and a is the last character of this prefix.) The ℓ_i-values are determined in the same way as the length-values of the matching statistics, introduced in [Chang 1994]. Using the suffix link table, the ℓ_i-values can be computed in $O(m)$ time altogether (cf. [Abouelhoda 2004]).

We now consider the relation of matching stems of two neighboring suffixes P_{i-1} and P_i for some $i > 0$. First note that $\ell_{i-1} \leqslant \ell_i + 1$. Moreover, for each q, $1 \leqslant q \leqslant \ell_{i-1}$ we have

$$\mathit{occ}_S(P[i-1..i+q-2]) \oplus 1 = \varphi(\mathcal{M}_{i-1}^q) \oplus 1 \subseteq \varphi(\mathcal{M}_i^{q-1}) = \mathit{occ}_S(P[i..i+q-2]) \tag{4.1}$$

where $M \oplus y = \{x + y \mid x \in M\}$ denotes the elementwise addition for any set M. That is, any suffix in $\mathcal{M}_{i-1}^q$ can be found in $\mathcal{M}_i^{q-1}$ with offset one.

To allow differences in our matching heuristic, we introduce the concept of matching branches which branch off from sets of the matching stem. We describe the branching in terms of a transformation of some suffix interval $\mathcal{M}_i^q$.

Let $i, 0 \leqslant i \leqslant m-1$ be arbitrary but fixed. Let q be such that $i + q - 1 < m$. Consider some suffix interval $\mathcal{M} = [l..r, h]$ such that the unit edit distance of $S[\mathsf{suf}[l]..\mathsf{suf}[l]+h-1]$ and $P[i..i+q-1]$ is exactly $d \leqslant k$. Then, for the edit operations $x \in \{\mathsf{MM}, \mathsf{I}, \mathsf{D}\}$, we define the matching branch

$Br_i(\mathcal{M}, q, d, x)$ as follows:

$$\begin{aligned} Br_i(\mathcal{M}, q, d, \mathsf{MM}) = \{(\mathcal{M}', q+1, d+1) \mid\ & d+1 \leqslant k, \\ & i+q < m, \\ & a \in \Sigma_{\mathrm{DNA}} \setminus P[i+q], \\ & \mathcal{M}' \text{ is the } a\text{-child of } \mathcal{M}\} \end{aligned}$$

$$\begin{aligned} Br_i(\mathcal{M}, q, d, \mathsf{I}) = \{(\mathcal{M}, q+1, d+1) \mid\ & d+1 \leqslant k, \\ & i+q < m\} \end{aligned}$$

$$\begin{aligned} Br_i(\mathcal{M}, q, d, \mathsf{D}) = \{(\mathcal{M}', q, d+1) \mid\ & d+1 \leqslant k, \\ & a \in \Sigma_{\mathrm{DNA}}, \\ & \mathcal{M}' \text{ is the } a\text{-child of } \mathcal{M}\} \end{aligned}$$

Any computation of a triple $(\mathcal{M}', q', d+1)$ according to these equations is called branching step. The MM-branching step implies a mismatch of $a \neq P[i+q]$ (in the reference sequence) with $P[i+q]$ (in the read). The I-branching step implies an insertion of character $P[i+q]$ in the read. The D-branching step implies a deletion of character $a \in \Sigma_{\mathrm{DNA}}$ in the read.

Note that in case some a-child of $\mathcal{M}$ does not exist, there is no corresponding contribution to the matching branch. We combine the different types of matching branches by defining:

$$Br_i(\mathcal{M}, q, d) = Br_i(\mathcal{M}, q, d, \mathsf{MM}) \cup Br_i(\mathcal{M}, q, d, \mathsf{I}) \cup Br_i(\mathcal{M}, q, d, \mathsf{D})$$

Obviously, any element in $Br_i(\mathcal{M}, q, d)$ can itself be extended by branching from it. To define this, we introduce for all $j \geqslant 1$ the iterative matching branch $Br_i^j(\mathcal{M}, q, d)$ as follows:

$$Br_i^j(\mathcal{M}, q, d) = \begin{cases} Br_i(\mathcal{M}, q, d) & \text{if } j = 1 \\ \bigcup_{(\mathcal{M}', q', d') \in Br_i(\mathcal{M}, q, d)} Br_i^{j-1}(\mathcal{M}', q', d') & \text{otherwise} \end{cases}$$

This gives us the matching branch closure $Br_i^*(\mathcal{M}, q, d)$, defined by

$$Br_i^*(\mathcal{M}, q, d) = \bigcup_{j \geqslant 1} Br_i^j(\mathcal{M}, q, d)$$

That is, $Br_i^*(\mathcal{M}, q, d)$ is the set of matching branches that can be derived by one or more branching steps from $(\mathcal{M}, q, d)$ (Fig. 4.4).

Of course, since each step increases the difference value d, the number of steps is limited by $k-d$. Each element $(\mathcal{M}', q', d') \in Br_i^*(\mathcal{M}, q, d)$ is extended by exactly matching $P[i+q'..m-1]$ against the enhanced suffix array beginning at the suffix interval $\mathcal{M}'$. That is, we compute $\mathsf{E}(\mathcal{M}', P[i+$

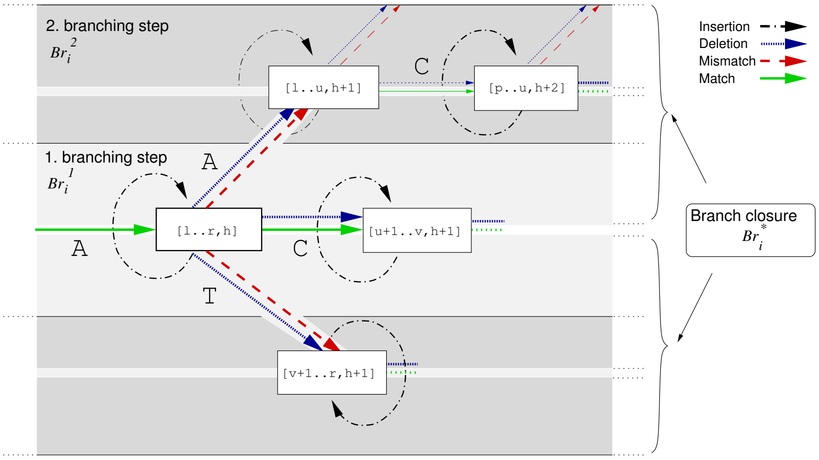


Figure 4.4: **The branch closure.** The suffix interval $[l..r, h]$, representing some string $w \in \Sigma^*_{\mathrm{DNA}}$ of length h, is split into its children $[l..u, h+1]$, $[u+1..v, h+1]$ and $[v+1..r, h+1]$ by matching an additional character $a \in \{\mathtt{A}, \mathtt{C}, \mathtt{T}\}$. We proceed building $\mathcal{M}_i$ by matching the character $\mathtt{C}$ (solid bold green line). Beforehand, alternative suffix intervals are stored in Br_i^1, either representing mismatches (dashed red line), insertions (dashed dotted black line) or deletions (dotted blue line). Br_i^2 holds suffix link intervals that in turn branch off from Br_i^1. The branch closure Br_i^* holds all such alternative intervals.

$q'..m-1]$).

While we have defined matching branches for any element in a matching stem, we only compute them for a few elements of the matching stem which make up the matching backbone: Let $Bb(P) = \{\mathcal{M}_i^q \mid 0 \leqslant i \leqslant m-1, q_i^{\min} \leqslant q \leqslant \ell_i\}$, where $q_i^{\min}$ is defined by

$$q_i^{\min} = \begin{cases} 0 & \text{if } i = 0 \\ \ell_{i-1} + 1 & \text{otherwise} \end{cases}$$

Thus, for each suffix i, $q_i^{\min}$ is the position in P from which to continue processing the next suffix. For any $\mathcal{M}_i^q \in Bb(P)$, we compute $Br_i^*(\mathcal{M}_i^q, q, 0)$. That is, we omit computing $Br_i^*(\mathcal{M}_i^q, q, 0)$ for $q < q_i^{\min}$. This is due to the fact that some of the suffixes in $\mathcal{M}_i^q$ are already included (with offset one) in $\mathcal{M}_{i-1}^q$, see equation (4.1). All in all, we arrive at a set $Q(P, k)$ of 4-tuples $(i, \mathcal{M}, q, d)$ such that the unit edit distance of $P[i..i+q-1]$ and w is $d \leqslant k$ and $\mathcal{M}$ is the suffix interval for w. The Algorithm 15 gives pseudocode for computing $Q(P, k)$ (which includes the matching backbone).

Turning to the analysis of the algorithm, first note that

$$|Bb(P)| \;=\; \ell_0 + 1 + \sum_{i=1}^{m-1} (\ell_i - \ell_{i-1} + 1) \leqslant m + 1.$$

That is, the matching backbone contains at most $m + 1$ elements and thus the statements in the inner loop of the algorithm (Alg. 15) are executed $O(m)$ times altogether. Obviously $Br_i(\mathcal{M}, q, d, \mathsf{MM})$ contains up to 5 elements, $Br_i(\mathcal{M}, q, d, \mathsf{I})$ contains at most 1 element and $Br_i(\mathcal{M}, q, d, \mathsf{D})$ contains at most 6 elements. Since there can be k iterations when computing $Br^*_i(\mathcal{M}^q_i, q, 0)$, the size of this set is at most $(12)^k$. Hence the total number of all matching branches is $(m + 1) \cdot (12)^k$. Each matching branch is generated from a previously generated element in constant time. Hence the algorithm runs in time proportional to $(m + 1) \cdot (12)^k$.

Algorithm 15 enumQ: enumerating a set Q of triples representing local alignments of P and S with at most k differences

```
Require: enhanced suffix array of S$, pattern P, difference threshold k
compute $\ell_i$ for any $i$, $0 \leqslant i \leqslant m-1$
compute $q_i^{min}$ for any $i$, $0 \leqslant i \leqslant m-1$
$Q \leftarrow \emptyset$
for $i \leftarrow 0 \ldots m-1$ do
  for $q \leftarrow q_i^{min} \ldots \ell_i$ do
    $Q \leftarrow Q \cup \{(\mathcal{M}_i^q, q, 0)\}$
    $U \leftarrow \{(\mathcal{M}_i^q, q, 0)\}$
    while $U \neq \emptyset$ do
      let $(\mathcal{M}, q, d) \in U$
      $U \leftarrow U \setminus \{(\mathcal{M}, q, d)\}$
      if $d > 0$ then
        $(\mathcal{M}', h) \leftarrow E(\mathcal{M}, P[i+q..m-1])$
        $Q \leftarrow Q \cup \{(\mathcal{M}', q+h, d)\}$
      if $d < k$ then
        $U \leftarrow U \cup Br_i(\mathcal{M}, q, d)\}$
```

From the matching backbone and from the set of all matching branches we select an element achieving a maximum score according to a simple scoring scheme where a character match scores +1 and a mismatch, an insertion and a deletion scores −1. The maximum score element $(i, [l..r, h], q, d)$ defines a set of substrings of S which are aligned to P. More precisely, for any j, $l \leqslant j \leqslant r$, P is matched against the reference substring $S[\mathsf{suf}[j] - (i + k)..\mathsf{suf}[j] + (m - i + k)]$ using the bit vector algorithm of Myers [Myers 1999]. For this, we allow a maximum number $r = \lceil 1 - \frac{a}{100} \rceil \cdot m$ of differences, according to the the identity threshold a. Myers algorithm runs in $O(m/\omega \cdot \ell)$ time where $\ell = 2k + m + 1$ is the length of the reference substring and ω is the word size of the machine. As $\omega = 64$ in our implementation, for reads of size up to 64, we have $m/\omega = 1$ and so the algorithm runs in $O(m + k)$ time. Note that this running time is independent of a. In summary, by specifying k along with some E-value [Karlin 1990] we set the thresholds to search for local alignment seeds. Subsequently, we use Myers algorithm to discards all seeds that produce poor semi-global alignments, according to parameter a, typically loosely set to values around 80% (which corresponds to $r = \lceil 1 - \frac{80}{100} \rceil \cdot m = \lceil 0.2m \rceil$).

Paired end and mate pair mapping

As briefly described in the introduction of this work, the term *mate pair* refers to two sepearate DNA fragments that have a know approximate distance on the orignal input DNA. The so-called insert size, i.e. the length of the DNA between the two fragments, varies between 1kb to 5kb. *Paired ends* refers to the sequencing of the same DNA molecule from two ends. The insert size thus depends on the length of the DNA molecules that are sequenced and the read lengths. Typically, for paired ends insert sizes of not more than a few hundred nucleotides are observed.

Using the algorithm described above a simple but effective mapping strategy for paired reads is employed. The mapping algorithm is first only applied to one of the ends. The seeds that are calculated using the enhanced suffix array are subsequently used to search the paired end or mate pair read using Myers bitvector algorithm (see chapter 2). The major differences between mate pairs and paired end are their expected distances on the reference genome and their orientation with respect to forward and reverse strand. The range for this alignment can be limited by the maximum expected insert size and the read length, typically <5000nt in any case. Due to its efficiency the bit vector algorithm is able to swiftly scan within these distances for the optimal alignment of the second read in both upstream and downstream direction on the forward and the reverse strand. A mate pair alignment is accepted if the identity threshold a criterion for the optimal alignment in this range is met. If no such alignment was found, the mapping algorithm in the enhanced suffix array is triggered also for the second read. If a heuristic alignment for the second read was found the bit vector algorithm is triggered a second time to search for the first read in the vicinity of the heuristic alignment. If the first read can be found within the identity threshold constraint, this pair will be reported. Otherwise, both reads will be flaged as unproper pair. This is a significant property as improperly mapped mate pairs and paired ends may indicate structural variation. Thus, this strategy reduces the number of false positive discordantly aligned reads. The pseudo-code for the mate-pair/paired-end mapping method is given in Alg. 16. Assume SgFwdRevAlign is a function that returns the first best semiglobal alignment on the forward and the reverse strand. Note, that in case the strandedness of the second read is known, e.g. in paired-ends, the function first aligns to the expected strand.

Algorithm 16 A matching strategy for paired ends and mate pairs

```
Require: ESA of S$, S, pattern $P_1$ and $P_2$ with maximum differences $r_1$
and $r_2$, minEvalue, identity threshold $k$, maximum insert size $\Delta$
$Q_1 = \mathsf{enumQ}(ESA, P_1, S, k)$
for $q \in Q$ do
  if score based evalue of $q$ statifies minEvalue then
    get alignment position $i$ from $q$ in $S$
    $\mathcal{A}_1 = \mathsf{SgFwdRevAlign}(P_1, S[i - r_1 \cdots i + |P_1| + r_1])$
    if $\mathbb{L}(\mathcal{A}_1) < r_1$ then
      $\mathcal{A}_2 = \mathsf{SgFwdRevAlign}(P_2, S[i - r_2 - \Delta \cdots i + |P_1| + r_2 + \Delta])$
      if $\mathbb{L}(\mathcal{A}) < r_2$ then
        report pair $\mathcal{A}_1, \mathcal{A}_2$
if no pair found then
  $Q_2 = \mathsf{enumQ}(ESA, P_2, k)$
  repeat for-loop above for $Q_2$ to obtain $\mathcal{A}_3$ and $\mathcal{A}_4$
  if no pair found then
    if $\mathbb{L}\mathcal{A}_1 < r_1$ then
      report single $\mathcal{A}_1$ for $P_1$
    if $\mathbb{L}\mathcal{A}_3 < r_2$ then
      report single $\mathcal{A}_3$ for $P_2$
  else
    report pair $\mathcal{A}_4$, $\mathcal{A}_3$
```

Primer and poly-A clipping and detection

In the case of cDNA sequencing reads are “contaminated” by primer and/or poly-A sequences at their 3′-end. These reads may not be aligned properly if the trailing contamination is not clipped prior to the alignment. In the following a strategy is presented to reconstruct potential primer sequences using a k-mer counting and assembly algorithm. Informally, the algorithm samples a number 5′-ends with a minimum sequence entropy of a constant length from the input. Windows of a constant size within the $3' - ends$ are mapped to integers and counted. The list of the most frequently occuring k-mers is subsequently assembled using a greedy chain algorithm. The algorithm is explained in detail.

For a number of input reads P of length m a sliding window of width k is shifted over the last e characters $P[m - e \cdots m]$. Each sequence within the window of width k, the k-mer, is stored. Sampling substrings form the 3′-ends of reads has the risk of accumulating low complexity sequences such as poly-A tails. To overcome this the shannon entropy [Shannon 1951] of each window is calculated.

Definition 20 (shannon entropy) *The shannon entropy is a measure for the information content of a realization of a random variable called 'message'. For some sequence S over the alphabet Σ and a probability function $p : \Sigma \to \mathbb{R}$ such* *that* $p(a), a \in \Sigma$ *is the probability of a character* a *occuring in* S*, the shannon entropy* H *is given by*

$$H(S) = -\sum_{a \in |\Sigma|} p(a) \cdot \log_2 p(a) \tag{4.2}$$

For equiprobable characters from the DNA alphabet we have a maximum the shannon entropy that evaluates to $4 \cdot 0.25 \log_2 0.25 = 2$. Note that the shannon entropy used in this context is a simple way to identify homopolymers and simple repeats. For repeats like $(ACTG)^n$, i.e. n repitions of a series of all four DNA alphabet characters, the shannon entropy would attain a maximum value too. Although this is in theory undesirable sequences like this are not likely to occur frequently at the ends of reads and so the shannon entropy suffices as a measure of sequence complexity. In this context the shannon entropy is used as a filter. Note that in the actual implementation all characters other than A, C, T, G, e.g. N's, are ommited and all sequences with $H(S) < 1.6$ are rejected.

Subsequently, a bijective mapping to the natural numbers I, i.e. a mapping that assigns a unique integer to each k-mer is achieved by using the fact that sequences can be interpreted as a positional number system to the base of $|\Sigma| = b$.

$$I(S) = \sum_{i}^{m} val(K[i]) \cdot b^i \tag{4.3}$$

In the context of DNA read k-mers we chose $b = 5$ to account for all four nucleotides and the undefined character N. The function val assigns a value from 0 to 4 to each character of Σ. For each k-mer K encountered during the iterations $C[I(K)] += 1$ is updated. Thus the array C holds the number of occurences for each K. At the same time we keep track of the most frequently occuring k-mers in array B. After the sampling step we aim to assemble the most frequent k-mers. First, two matrices of $3'$- and $5'$- right and left overlaps is constructed for each pair K_1, K_2 of k-mers.

Definition 21 (longest overlaps) *The longest* **right overlap** *problem for two sequences* S *and* T *is to find a* k *such that* $S[n-k-1 \cdots n-1] = T[0 \cdots k-1]$ *and* $S[n-k-2 \cdots n-1] \neq T[0 \cdots k]$. *In other words, the longest match of a suffix of* S *with a prefix of* T *The longest* **left overlap** *is analogously the longest match of a suffix of* T *with a prefix of* S.

Informally, the longest right overlap can be found by simply scanning the sequence T from left to right using the following short algorithm (Alg. 17). Note, that Alg. 17 could report a full overlap of two sequences. Due to the encoding of the sequences this is not possible in the presented scenario.

The algorithm obviously runs in $O(n)$ and has a constant space requirement for each pair S and T. Calculation of the left overlap requires

Algorithm 17 Rovl : Finding longest right overlap of S and T

```
**Require:** $S$ of length $n$ and $T$ of length $m$
$i = \min(n-1, m-1)$
for $j := n-1$ to $0$ do
  if $T[j] = S[i]$ then
    $i = i-1$
  else
    $i = n-1$
return $(n-1) - i$
```

only to swap the input sequences. Finally, we have stored the left and right overlap in two matricies. Now we proceed building one left and one right chain for a number of the most frequently occuring k-mers. For a given chain we greedily select the best k-mer with the best overlap and mark this k-mer not to be used again. The greedy chaining part runs in quadratic time with respect to the number of the selected k-mers. This number is upper-bounded by $maxkmers$ which limits the runtime. The algorithms complexity is further reduced (Alg. 18) because the chains are required to incorporate the $maxchains$ most frequent k-mers, where $maxchains < maxkmers$.

It now remains to clip the primer sequences. It is important to note that the primers, wheter they are autmoatically detected or explicitly given to the program, may not be fully present at the $3'$-end. In many cases only the $5'$-end of the primer was actually sequenced. Furthermore, increasing error rates, especially in close vicinity to error prone homopolymer signals, a more sensitive alignment strategy is required for clipping. Informally, if a poly-A signal and/or a $3'$-primer needs to be clipped, a clip sequence C is constructed. Note that the clip sequence can be longer than the acutal read. Therefore, in this work, a local alignment strategy is chosen. The accumulation of mismatches, insertions and deletions in the alignment of R with C, especially in the poly-A sequence, disagrees with the hypothesis that sequence is post-transcriptional and not genomic. Thus mismatches and indels are penalized with -2 score points while matches are rewarded with only 1 score point. If poly-A signal and primer need to be clipped at the same time, a poly-A sequence of the length of the read r is extend by the primer sequence. An local pairwise sequence alignment of C with the read r is deemed successful if the following two criteria are met

1. the best local alignment has a minimum score
2. the end of the best local alignment is no more than w nucleotides away from the $3'$-end of r

A successful alignment leads to the clipping of the read. Otherwise the

Algorithm 18 Automatic 3′-Primer assembly

```
Require $\mathit{maxkmers}$, $\mathit{maxchains}$, set of reads $\mathbb{A}$, k-mer size $k$, 3′-end $e$
for $R \in \mathbb{A}$ do
  for $i = e$ to $m - k - 1$ do
    if $H(P[i \cdots i + k]) > 1.6$ then
      $C[I(P[i \cdots i + k])] += 1$
$B = \mathsf{sort}(C)[0 \ldots \mathit{maxkmers}]$
for $\forall S, T \in B$ do
  $R[S, T] = \mathsf{Rovl}(S, T)$
  $L[S, T] = \mathsf{Rovl}(T, S)$
for $i = 0$ to $\mathit{maxchains}$ do
  mark all $B$ unused
  $\Pi_i = B[i]$
  mark $B[i]$ used
  $j = 0$
  while new k-mer found do
    $q_{max} = \arg\max_q R[\Pi_i[j], B[q]]$ and $B[q_{max}]$ unused
    $\Pi_i = \mathsf{chain}(\Pi_i, B[q_{max}])$
    mark $B[q_{max}]$ used
    $j = j + 1$
  $j = 0$
  while new k-mer found do
    $q_{max} = \arg\max_k L[\Pi_i[j], B[q]]$ and $B[q_{max}]$ unused
    $\Pi_i = \mathsf{chain}(\Pi_i, L[q_{max}])$
    mark $B[q_{max}]$ used
    $j = j - 1$
return longest chain $\Pi$
```

read is mapped unclipped to the reference.

4.3 Results

`segemehl` constructs indices either for each chromosome of a genome and the matching is performed chromosome-wise or, depending on the available RAM, chromosomes are combined to larger sequences. Compared to other methods, the index structure used by `segemehl` is significantly larger. For example, the enhanced suffix array of human chromosome 1 occupies approximately 3 GB of space. As it is stored on disk, the index only needs to be computed once. The construction of the index requires linear time. For example, on a single CPU, the construction of the complete enhanced suffix array for human chromosome 1 takes approximately 15 minutes. For our comparisons, we ran `segemehl` with maximum occurrence parameter $t = 500$. The maximum E-value for seeds was set to 0.5 and minimum identity threshold to $\alpha = 85\%$ which corresponds to a maximum of $\lceil 0.15 \cdot m \rceil$ differences in an alignment of the read of length m.

In the initial publication we compared `segemehl` to `Bowtie` v0.9.7 with option `-all`, `BWA` v0.2.0, `MAQ` v0.7.1, `PatMaN` v1.2.1 and `SOAP` v1.11 with option `-r 2`. `MAQ` and `SOAP` are based on ungapped alignments which are computed by hash lookups [Li 2008a, Li 2008b, Li 2009a]. Due to length restrictions, `MAQ` is limited to Illumina (and SOLiD) reads. It additionally takes quality scores into account. The quality values needed by `MAQ` were, for all nucleotides, uniformly set to a value corresponding to the error rate. `Bowtie` [Langmead 2009] and `BWA` [Li 2009a] index the reference genome with the Burrows-Wheeler transform. `BWA` allows a limited number of indels. `PatMaN` [Prüfer 2008] matches the reads by traversing a non-deterministic suffix automaton constructed from the reference genome. Except for `PatMaN`, all programs only report the matches with the smallest edit distance. `BWA` and `Bowtie` each need about 10 minutes to build their index. The fastq files needed by `MAQ` are built in approximately 2 minutes. `PatMaN` and `SOAP` require no indexing steps. The options for the other programs were chosen so as to achieve results similar to `segemehl`. For our comparison, we performed tests on simulated as well as real-life read data sets. For the simulation we generated read sets representing different error rates, types and distributions. We used three distinct error sets, one containing only mismatches, one containing only indels and a last one representing reads with mismatches and indels at a ratio of 1:1. Additionally, different error distributions were used to model error scenarios such as terminal contamination (e.g. linker, poly-A tails) or decreasing read quality. We chose uniform, 5′, 3′ and terminal error distributions.

Each simulated dataset contained 500 000 simulated reads, each of length 35 bp, sampled from a 50 MB large region of the human genome (chromosome 21). We introduced errors to each simulated read according to previously defined rates, error types and distributions. For the 50 MB region we constructed the indexes required for `segemehl` and `Bowtie`. For `MAQ` we constructed the index for the read set under consideration. Index construction took approximately one minute for `Bowtie` and `BWA`. The construction for the enhanced suffix array for `segemehl` took 3.5 minutes. The binary fastq files for `MAQ` were created in about 20 seconds.

We ran `segemehl` with seed differences threshold $k = 0$ and $k = 1$. For $k = 0$, only exact seeds are computed and for $k = 1$ seeds with at most one difference are computed. All programs were executed single-threaded on the same machine. The results for a uniform error distribution for mismatches only as well as for mismatches and indels are shown in Fig. 4.5. We measured the performance in terms of running time (Fig. 4.5 (A)) and recall rates, i.e. the percentage of reads mapped to the correct position. `segemehl` has recall rates of more than 95% ($k = 1$) and 80% ($k = 0$) in

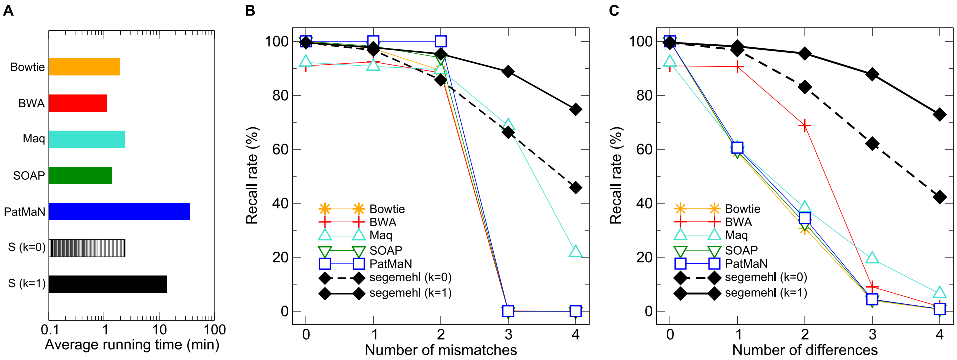


Figure 4.5: **Comparison of recall rates and running time for several short read aligners.** Average running time for the different programs (A) in matching runs with 500 000 reads in two different data sets (logarithmic scale; S refers to `segemehl`). The differences are uniformly distributed and consist of only mismatches (B) or mismatches, insertions and deletions (C). The recall rate describes the fraction of reads which was mapped to the correct position. All programs were used with default parameters. `Bowtie` was called with option `-all` and `SOAP` with option `-r 2`.

each setup with not more than two errors in the reads. With four uniformly distributed errors in the reads, the recall rate drops below 80% ($k = 1$) and 50% ($k = 0$), respectively. Hence, for $k = 1$ `segemehl` outperforms all other methods in terms of recall rates. For reads containing only mismatches and $k = 0$, `segemehl` is comparable to other methods (Fig. 4.5 (B)) while it has a significantly better recall rate as soon as insertions and deletions are involved (Fig. 4.5 (C)). As expected, the recall rate of most short read aligners drops if insertions and deletions are introduced into the reads. The running time of `segemehl` for $k = 0$ is comparable to other short read aligners. For $k = 1$, the running time increases by a factor of 10.

In contrast to `Bowtie`, `BWA`, `MAQ`, and `SOAP`, `segemehl` reports, by default, multiple matches for a read within the reference genome if the corresponding alignments have an E-value smaller than some user defined threshold. This behavior leads to an increase in the running time and a decrease in specificity. Compared to `PatMaN`, which is also able to report multiple matches, `segemehl` can cope with more than two differences and still is on average faster by a factor of 1.7 ($k = 1$) and 14 ($k = 0$). As expected, the worst `segemehl` results are seen for high error rates with a uniform error distribution (Fig. 4.6). Terminal, $3'$ and $5'$ error distributions yield better results, suggesting that `segemehl` implements a robust method that is insensitive to leading and trailing contaminations.

Next, we compared `segemehl`, `Bowtie` and `MAQ` on two real-life data sets. We used `Bowtie` with option `-all` and `MAQ` with option `-C 513` as suggested in the manuals to achieve maximum sensitivity. `segemehl`'s sensitivity was controlled by option `-M 500` to omit all

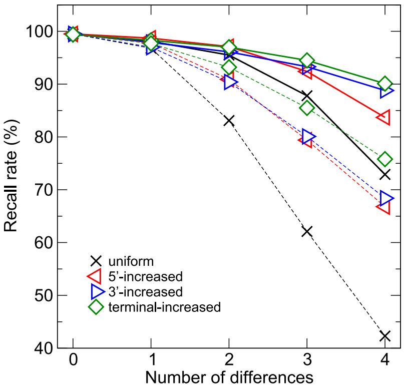


Figure 4.6: **segemehl recall rates for varying difference values and distributions.** Recall rates are depicted for $k = 0$ (dashed) and $k = 1$ (solid). For terminal–, 3′– and 5′–increased difference distributions, segemehl achieves a recall rate above 80% for reads with 4 errors.

seeds occurring more than 500 times in the reference sequence. The data set ERR000475 of 20 million Illumina reads (length 45) for *H. sapiens* was downloaded from the NCBIs Short Read Archive (http://www.ncbi.nlm.nih.gov/Traces/sra/). The second data set comprised about 40 000 short 454 reads from the *arabidopsis mpss plus database* (http://mpss.udel.edu/at/). The average length of the 454 reads was 23 bp. We partitioned the 454-set into subsets of equal size, to satisfy input requirements for MAQ. An average quality value was assigned to each base. Mapping multiple reads to a reference genome is a task which can easily be parallelized. Like all other methods, segemehl offers a parallelization option to run the program on multiple cores. segemehl runs for the ERR000475 dataset were carried out in eight parallel threads on a single machine with two Quad-core CPUs and 16GB of RAM. Seven enhanced suffix arrays were constructed representing the whole human genome. segemehl mapped 92% of the reads to the reference sequence while MAQ mapped 85% without and 89% with quality values. The corresponding values for Bowtie are 81% and 89%. The largest difference between the three tools is for the total number of exact matches. Although MAQ was, according to the manual, running in maximum sensitivity mode, segemehl computes 20 times more matches than MAQ (Tab. 4.1 (a)). Bowtie reports 2.5 billion matches which is much more than the two other tools. As expected, for the 454-set, the difference among the compared programs is even larger. While Bowtie is able to map only 71% of all reads, segemehl achieves 95%. MAQ, a program explicitly designed for Illumina reads, matches 79% of the reads. Interestingly, compared to Bowtie, MAQ reports more matches with two mismatches. segemehl mainly achieves this result by mapping more reads with one or two er-

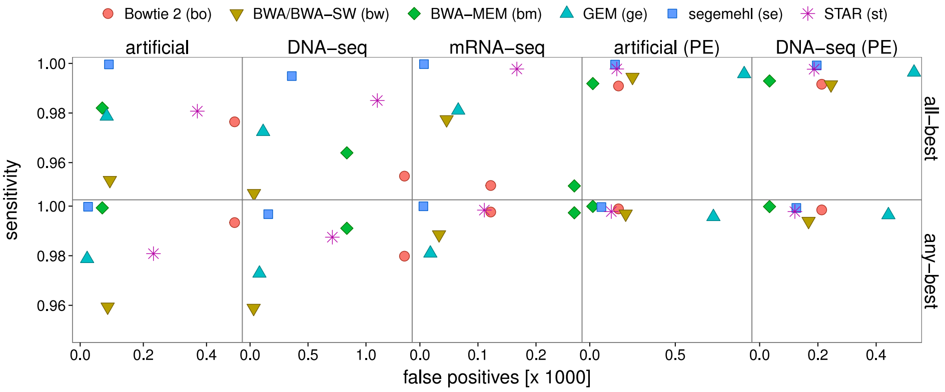


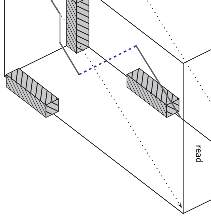
Figure 4.7: **Benchmark updates from [Otto 2014].** The performance is assessed in terms of sensitivity and false positives. `segemehl` performed better with respect to sensitivity and number of false positive alignments than most of the other tools with default parameters at the cost of higher running times.

rors. In fact, by allowing insertions and deletions `segemehl` doubles the number of reads matched at the unit edit distance of 1 (Tab. 4.1 (b)). Since the initial publication several new tools have emerged while the support of others, such as Maq, has been discontinued. Therefore, the benchmarks of this mapping model have been updated in [Otto 2014]. Specifically, we evaluated `segemehl` with BWA [Li 2009a], BWA-SW [Li 2010a], STAR [Dobin 2013a], GEM [Marco-Sola 2012], Bowtie2 [Langmead 2012b] and BWA-MEM [Li 2013].

For the benchmarks on simulated data, the reads were simulated using Mason [Holtgrewe 2011b] with default error models. In contrast to the prior benchmarks we have additionally evaluated the sensitivityand specificity of `segemehl` and its competitors on *real* data. To do this we sampled reads from various DNAseq and RNAseq data sets and aligned them with RazorS [Weese 2012] in full sensitivity mode and determined the optimal alignment with respect to the unit edit distance. Specifically, we evaluted the tools with reads from the publically available DNAseq (ERR037900) and RNAseq (SRR534289) data. 454 data was sampled from the data set with accession SRR003161. To account for the different strategies to deal with multiple optimal alignments two different benchmark setups were chosen. irst, we measured the sensitivity and the number of false positive alignments (FP) for each program in finding at least one optimal hit (any-best) with respect to the unit edit distance. The second benchmark measured the performance in finding all optimal hits (all-best). The evaluations on simulated and real data shows that `segemehl` compares favourably in with other tools. In virtually all cases segemehl, especially for the all-best scenario, achieves the highest sensitivity. At the same time it maintains one of the highest specificities (Fig. 4.7). Thorough and reproducible updated benchmarks are described in [Otto 2014].

				mismatches + insertions + deletions			
			total	0	1	2	$\geqslant 3$
(a) Human genomic data set ERR000475 (Illumina)							
	`Bowtie` (-all)	all matches	2 692 341 844	631 194 732	925 094 123	1 136 952 989	-
		reads found	16 011 867 (81%)	12 006 627	2 824 359	1 180 881	-
	`Bowtie` (-all) with qualities	all matches	9 264 604 839	631 194 732	914 965 615	1 098 260 521	6 620 183 971
		reads found	17 693 135 (89%)	12 006 627	2 806 842	1 162 905	1 716 761
	`MAQ`	all matches	67 108 174	22 545 585	15 999 878	14 913 062	13 649 649
		reads found	16 762 361 (85%)	12 006 627	2 829 601	1 199 110	727 023
	`MAQ` with qualities	all matches	96 980 574	15 084 354	9 867 729	10 987 486	61 041 005
		reads found	17 725 314 (89%)	11 277 928	2 928 839	1 364 477	2 154 070
	`segemehl`	all matches	701 943 169	464 294 770	112 471 308	42 794 605	57 262 900
		reads found	18 191 858 (92%)	12 002 123	2 872 615	1 221 313	2 095 807
(b) *A. thaliana* short RNA data set (454)							
	`Bowtie` (-all)	all matches	156 621	85 254	42 443	28 924	-
		reads found	26 969 (71%)	18 739	5 390	2 840	-
	`MAQ`	all matches	74 994	26 890	15 078	14 482	18 544
		reads found	29 987 (79%)	18 738	5 389	3 093	2 767
	`segemehl`	all matches	262 262	72 328	83 070	51 048	55 816
		reads found	35 942 (95%)	18 737	10 525	3 744	2 936

Table 4.1: Comparison of the performance of `Bowtie`, `MAQ`, and `segemehl` on two real-life datasets. (a) The genomic paired DNA library with 19 812 604 Illumina reads was matched using `MAQ` (chromosome by chromosome), `Bowtie` (single index), and `segemehl` (seven enhanced suffix arrays each representing a disjoint subset of the human chromosomes). `Bowtie` was used with and without quality values. To simulate a `MAQ` run without quality information, an average quality value was assigned to all bases of the Illumina data set. The total number of matches differs significantly: `Bowtie` outnumbers all other programs. `segemehl` still reports ten times more matches than `MAQ` without quality values. The number of exact matches shows a 20-fold increase. Although `MAQ` improves when quality values are used, the total number of matches remains small in contrast to the other programs. Differences in the number of exactly matching reads reflect the distinct handling of repetitive and uninformative reads. In `segemehl`, all reads matching more often than $t = 500$ times are dropped. (b) 38171 reads of a small short RNA library sequenced with 454 were matched to the *A. thaliana* genome. Compared to `Bowtie` and `MAQ`, `segemehl` mapped significantly more reads. Allowing for one error, `segemehl` matches twice as many reads as `Bowtie`, due to the fact that `segemehl`, unlike `Bowtie`, allows for indels. Note that `segemehl` discarded a few perfect matches since the corresponding seeds occur more than $t = 500$ times in the reference sequence.



Chapter 5

Split read mapping

Contents

5.1 Introduction 69
5.2 An algorithm for split read mapping 71
5.3 Results 75

High-throughput sequencing technologies allow the analysis of whole transcriptomes in yet unknown detail and quantity. Several studies have focused on detecting splice sites for conventionally spliced mRNAs in RNA-seq data. Likewise, the detection of fusion transcripts through gene fusion or trans-splicing has become an important focus in cancer research. Here, we introduce a novel algorithm to detect splicing, trans-splicing, and gene fusion events from short single-end RNA sequences in an unbiased fashion. In contrast to other methods, which are restricted to single exon-exon junctions, the proposed multi-split read algorithm accommodates more complex structures. Making use of the increasing read lengths in different sequence technologies it thus directly recovers larger parts of transcribed isoforms. It outperforms other approaches for splice site detection in terms of precision and speed while maintaining the same level of sensitivity. We show that the method's precision is largely invariant to sequencing errors. Our method reliably detects previously described trans-splicing events in *Drosophila melanogaster*, reports a massive amount of split reads mapping to the PMEL locus for a melanoma RNAseq data set and predicts previously undescribed splicing events. This chapter summarizes the work of [Hoffmann 2014a].

5.1 Introduction

The term splicing refers to a post-transcriptional process in which the raw transcript (pre-mRNA) is cleaved from intronic DNA fragments. In general, the splicing mechanisms allow the recombination of protein coding and non-coding RNA fragments and thus greatly increase the repertoire of potentially functional transcripts. While the overwhelming majority of splicing events occurs within the same pre-mRNA at consensus splice sites, some mRNAs are spliced at non-consensus sites. Many transcripts derived at non-consensus splice sites may have escaped detection in the past because of the assumptions built into the *in silico* analysis pipelines or due to the limited throughput of earlier RNA-seq protocols.

Some species have developed mechanisms to fuse separately transcribed mRNAs. These mRNAs may stem from distant loci, opposite strands, or homologous chromosomes. A prominent physiological example is the *mod(mdg4)* trans-splicing in *Drosophila melanogaster* [Dorn 2001]. Chimeric transcripts from different loci may be functional even in mammals [Frenkel-Morgenstern 2012b]. Circular RNAs [Jeck 2013, Salzman 2012] are recognizable in RNA-seq data in the form of reads that contain apparent splice junctions that connect the end (start) of a split read fragment to the start (end) of a downstream (upstream) fragment. Very recently, they have been identified as abundant class of regulatory transcripts functioning as microRNA sponges [Memczak 2013, Hansen 2013]. In addition to physiological trans-splicing, a number of transcripts potentially derived from gene fusions have been observed in different types of cancer such as melanoma [Berger 2010] or breast cancer [Edgren 2011]. In the following, we use for brevity the term "fusion transcript" to refer to RNAs that stem from a fused gene or a trans-splicing event. Although trans-splicing and mRNA fusion events appear to be rare compared to the regular local and co-linear splicing, fusion transcript point to potentially important functional entities or diagnostic marker genes. Their emerging importance mandates the use of analysis pipelines for RNA-seq data that ensure their efficient detection and inclusion in the subsequent data analysis work-flow.

Several different algorithms for splice site detection have been devised so far. The original version of `TopHat` [Trapnell 2009] predicts exon locations from the coverage data and attempts split read alignments across neighboring exons. This algorithm was not able to detect fusion events. To remedy this situation a new algorithm, `TopHat`-Fusion [Kim 2011a], was published and has since been integrated into `TopHat2` along with some other modifications to the original algorithm. `SpliceMap` [Au 2010] starts by splitting the reads into fragments of 25nt and then attempts to align all fragments separately with a limited number of mismatches. Subsequently, canonical splice junctions are searched within a genomic interval of 400MB. The specificity of splice junctions may be improved by providing paired-end information. Although, `SpliceMap`'s junction search is significantly distinct from `TopHat2`'s. The `MapSplice` algorithm [Wang 2010] resembles `SpliceMap`. It also performs a segmentation of reads into tags and handles each tag individually. The tags are aligned to exons and junctions inferred from tags mapping to consecutive exons.

`SplitSeek` [Ameur 2010] is also based on the idea of using both 5′-

and 3′-ends of a reads to infer spliced exons. `SplitSeek` does not make use of canonical splice site information and is not limited to a common locus. Another tool that was specifically designed for the detection of fusion transcripts, `deFuse` [McPherson 2011], makes use of paired-end information and triggers local alignments at positions of discordant paired-end reads.

Like `TopHat2`, `SOAPsplice` [Huang 2011] is based on a Burrows-Wheeler transform and attempts to map the reads completely to the genome with no more than 3 mismatches or one gap. All unmapped reads are subsequently subjected to a split mapping with two segments. Each segment has to fulfill a number of quality criteria. `GSNAP` [Wu 2010] is based on hash-tables to retrieve position lists and subsequently merges and filters them efficiently. It is able to allow for multiple mismatches and long indels and can detect short- and long-distance splicing. The RNA-seq mapper `RUM` [Grant 2011] uses `bowtie` [Langmead 2009] and `BLAT` [Kent 2002] to detect annotated as well as *de novo* splice junctions. In a first step, reads are mapped against the genome as well as the user-supplied transcriptome. All unmapped reads are forwarded to BLAT and split alignments are merged. One of the latest tools for RNA-seq alignment, `STAR` [Dobin 2013b], is based on maximal mappable prefixes (MMP) that are identified using suffix arrays. In a second step, the prefixes are "stitched" together to reconstruct the isoforms. This algorithm was reported to be very fast – in fact more than 50 times faster than some of its competitors.

Here, we present a unified, unbiased algorithm that allows the detection of splicing, trans-splicing, and gene fusion events from single-end read data. The method, based on enhanced suffix array, chaining, and dynamic programming algorithms, is integrated in the mapping tool `segemehl` [Hoffmann 2009].

5.2 An algorithm for split read mapping

The algorithmic strategy for split-read alignments is sketched in Figure 5.1.

Chaining

After matching a read we obtain at most $2m - \ell$ seed alignments. Each seed may have multiple occurrences in the genome. The start positions of each seed's set of alignments in the reference genome are represented by the ESA interval $[l, r]$. In addition to the aforementioned E-value and maximum occurrence parameters, the alignment seeds retrieved from the ESA are required to have a minimum Shannon entropy of 1.5. The Shannon

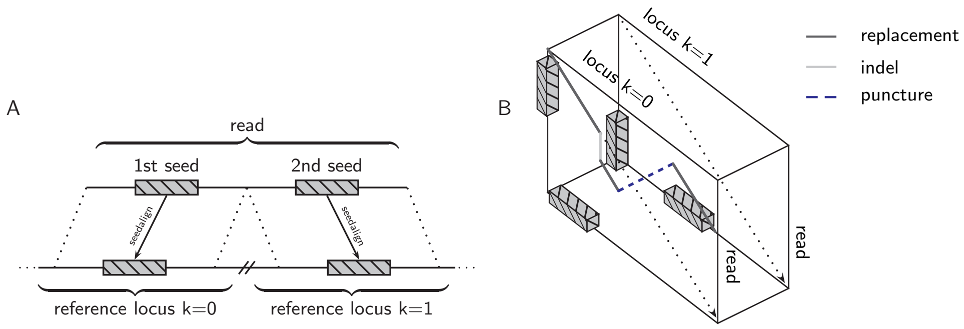


Figure 5.1: **A chain of seeds guides a local transition alignment acrossmultiple genomic loci.** High-quality seeds mapping to different genomic loci, strands or chromosomes (A) are chained. Subsequently, the order of the seeds within the chain guides a walk through the alignment cube (B). For each genomic locus, a local alignment with the read is performed. In addition to the regular Smith-Waterman recursions, the local transition alignment allows crossing between different reference loci.

entropy of a sequence S is defined by

$$H(S) = -\sum_{i=0}^{n-1} p(s_i) \log_2 p(s_i) \tag{5.1}$$

where $p(s_i)$ denotes the probability of the character s_i to occur in S. This additional prerequisite is necessary to drop low-complexity seeds caused, e.g., by poly-A tails or repeats that bypass the maximum occurrence threshold due to sequencing errors. In general, it can not be ruled out that the Shannon entropy filter affects the detection of splice events in repetitive elements (cf. [Lin 2008b]). However, our calculations show that the majority of 20bp and 40bp windows in human ALU repeats have a Shannon entropy well above our threshold of 1.5. Therefore, this filter does not impede the split-read mapping in ALU repeats per se. After passing the three filters, each alignment start of a seed in the reference genome is easily obtained in constant time using the suffix table of the ESA. Let $\mathcal{S}$ denote the set of seeds. In the chaining step, we aim to select an ordered chain of seeds $c \subseteq \mathcal{S}$ that optimally covers the read from start to end while at the same time maximizing the sum of alignment scores. Let ψ denote a function to obtain the alignment score of a seed and let π_s and π_e be two functions to determine a seed's alignment start and end in the read, respectively. Finally, the score of a chain is evaluated using

$$\sigma(c) = \sum_{k=1}^{|c|} \psi(c[k]) - \sum_{k=2}^{|c|} |\pi_e(c[k-1]) - \pi_s(c[k])| \tag{5.2}$$

where $c[k] \in \mathcal{S}$ denotes the k-th seed in chain c. In our implementation $\psi(c[k])$ is the number of correctly matched nucleotides of fragment i minus the sum of mismatches, insertions and deletions in this fragment. A set of chains is obtained using a greedy chaining algorithm (Algorithm 19). Initially, seeds are sorted with respect to π_s and the sorted list is stored in C. In the first step, each single seed is a chain of its own. The computation proceeds by iterating over all chains in the list C. For each chain C_i, the best preceding chain c' is identified and concatenated with it. For two chains c', c, the concatenation operator is denoted by $g(c', c)$.

Algorithm 19 A greedy chaining algorithm

```
$C = \mathrm{sort}(\mathcal{S}, \pi_s)$
for i:= 1 to |C| do
  $c' = \varepsilon$
  for j := 1 to i-1 do
    if $\sigma(g(C_j, C_i)) > \sigma(C_i)$ then
      $c' = C_j$
  if $c' \neq \varepsilon$ then
    $C_i = g(c', C_i)$
```

It is easy to see that algorithm terminates after $(|C| \cdot (|C| - 1))/2$ iterations. Since there are at most $2m - \ell$ seeds, the algorithm is of the complexity $O(m^2)$.

Local transition alignment

The chains are ranked with respect to their score. To ensure a high precision, only the highest ranking chain is used in the subsequent alignment procedure. Furthermore, the chain by default should cover more than 80% of the read. As pointed out above, each seed might have multiple alignments across the genome. In this case `segemehl` selects those alignments that minimize the distance on the genome and are -if possible- on the same strand. Finally, for each seed we obtain exactly one position in the reference. These positions guide a local transition alignment across a number of loci. A similar idea has been independently proposed by [McPherson 2011]. In contrast to McPherson, we devise an algorithm that fully integrates the transition between multiple matrices to obtain an optimal local split alignment across different genomic loci. The local transition alignment method is a modification of the Smith-Waterman alignment (Algorithm 20).

Let $\gamma(c[k])$ be a function that returns the sequence for the reference locus to which the seed $c[k]$ was aligned to. Note, that the sequence returned by γ extends the exact alignment boundaries of $c[k]$ by several nucleotides to account for inaccuracies of `segemehl`'s seed finding heuristics. The

Algorithm 20 A transition alignment algorithm across multiple loci

```
for $k := 1$ to $|c|$ do
  for $i := 1$ to $m$ do
    $lms[k,i] = lms[k,i-1]$
    for $j := 1$ to $|\gamma(c[k])|$ do
      a = getchar(r,i); b = getchar($\gamma(c[k])$,j)
      $M[i,k,j] =$
      $\max \begin{cases} M[i-1,k,j] + \delta \\ M[i,k,j-1] + \delta \\ M[i-1,k,j-1] + s(a,b) \end{cases}$
      for $q := 1$ to $k-1$ do
        $M[i,k,j] =$
        $\max \begin{cases} M[i,k,j] \\ lms[q,i-1] + \sigma(a,b) \end{cases}$
      $lms[k,i] = \max(M[i,k,j], lms[k,i-1])$
```

algorithm iterates over all seeds in the chain c and performs local alignments of the read with their respective reference locus. Insertions and deletions are penalized with δ. $s(a,b)$ scores matches and mismatches. The resulting alignment scores are stored in a 3-dimensional matrix M. During the local alignment at $\gamma(c[k])$ the algorithm keeps track of the last maximum score $lms[k,i]$ seen prior to the alignment of the i-th character of the read. This additional table is the key to the local transition alignment algorithm. In addition to the local alignment recursions that maximize the score of $M[i,k,j]$, all $k-1$ preceding loci are checked for a possible transition using the *lms*-table.

Simulations and Tools

For the simulation of both regular and irregular splice junctions, a sample of 10 000 isoforms was drawn from the ASTD data base [Koscielny 2009]. For the non-regular data set, 20% of the exons were either flipped to the opposite strand or substituted by a distant exon from ASTD data base where any exon with distance > 200kB from the isoform or on a different chromosome is denoted as long range splicing. The isoforms of each data set were extracted from the Human genome (hg19) by concatenating their exon sequences. Using mason v.0.1.1 [Holtgrewe 2011b], we simulated Illumina and 454-like single-end reads of length 100 nt and 400 nt, respectively, from the isoforms of each data set with 10-fold, 15-fold, and 20-fold coverage. The parameter values of the Illumina and 454 error model in mason were specified in accordance to the Bowtie2 paper [Langmead 2012a]. For simulated Illumina reads, we used the default parameters; for 454, we used `-k 0.3 -bm 0.4 -bs 0.1`. Recall and false positive rates were calculated on the basis of the splice junctions predicted by each tool. Thus, the recall was calculated as the fraction of simulated junctions correctly identified by each tool. For each tool, the false positive rate was calculated by dividing the number of wrong junctions by the number of predicted junctions. For the comparisons we have used `TopHat2` (version 2.0.4), `RUM` (version 1.12_01), `STAR` (version 2.1.3e_r157), `SOAPsplice` (version 1.9), `MapSplice` (version 2.1.2), `GSNAP` (version 2013-11-27), and `SpliceMap` (version 3.3.5.2 (55)). All programs have been run with default parameters for split read mapping. With the exception of `RUM` no additional annotation information was provided to any of the tools. Details are given in the Appendix.

5.3 Results

The algorithm presented here makes use of a read matching method with enhanced suffix arrays (ESA) described above [Hoffmann 2009]. In brief, for a read of length m, the algorithm evaluates the best alignments with a limited number of mismatches, insertions, and deletions for all $2m - \ell$ suffixes of the read of length m and its reverse complement, where ℓ is a minimum suffix length. An alignment qualifies as a seed if a score-based maximum E-value criterion and a maximum occurrence threshold are met. Subsequently, full reads will be aligned to all distinct seed locations in the reference genome using Myers' semi-global bit-vector alignment [Myers 1999]. All alignments passing a minimum accuracy threshold are reported. While the E-value, minimum accuracy, and maximum occurrence parameter control the specificity and limit the number of multiple hits, the large number of alignments from the beginning to the end of the read ensure a high sensitivity. In the case of spliced or fusion transcripts, a successful semi-global alignment of the read is likely to fail. Instead, the ESA-based method will identify several seeds matching to different locations or strands. The algorithmic strategy to identify splicing, trans-splicing, or gene fusion sites is based on a greedy, score-based seed chaining followed by a Smith-Waterman-like transition alignment.This alignment optimizes the total score of a number of local alignments at different locations and strands. The algorithm does not have any effective length restrictions. Details are given in the Methods section.

Simulated data

The algorithm's performance was compared with seven alternative split read methods: `TopHat2`, `RUM`, `STAR`, `SOAPsplice`, `MapSplice`, `SpliceMap`, and `GSNAP`. In principle, all tools were run with default parameters for split read mapping. Where available, fusion and trans-splice sensitive alignment parameters were turned on. With the exception of

`RUM`, no extra annotations have been given to any of the programs (see Appendix).

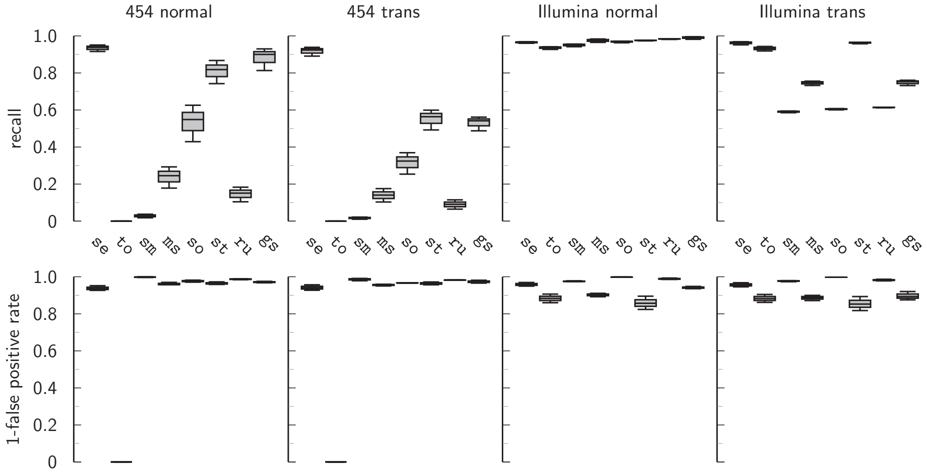


Figure 5.2: **Performance of various read aligners on simulated data sets with different splice events.** On simulated 454 reads (400bp), `segemehl` performs significantly better in detecting conventional and "non-conventional" (strand-reversing, long range) splice junctions. `segemehl` is the only tool that consistently recalls more than 90% of conventional splice junctions. For "non-conventional" splice events, `segemehl` extends its lead to 40% in recall – without losing precision. Likewise, compared to three of seven alternative tools `segemehl` shows a 30% increase of the recall for irregularly spliced Illumina reads (100bp). Compared to `TopHat2`, it shows a slight increase while reporting significantly fewer false positives. At the same time, `segemehl`'s performance on simulated, regularly spliced Illumina reads is comparable with the other seven tools tested. (`to: TopHat2; se: segemehl; sm: SpliceMap; ms: MapSplice; so: SoapSplice; st: STAR; ru: RUM; gs: GSNAP`)

To test the algorithms' precision and sensitivity in all applicable scenarios, Illumina and 454 reads were simulated for regular splice junctions and a mixture of regular and non-regular splice junctions, i.e., splice junctions that connect opposite strands (strand-reversing) and splice junctions that connect distant exons (long range splicing; see Methods). The simulated Illumina and 454 reads were 100bp and 400bp long, respectively. Furthermore, we tested the recall on short circular transcripts (100bp) as well as long linear and long circular reads of length 0.5-5kB. The latter is the typical size range for circular transcripts in mammals. Short circularized RNAs are abundant in Archaea [Danan 2012]. In accordance with other publications in this field, Illumina and 454 error models were applied to the simulated reads. These models introduce mismatches, insertions, and deletions to the reads in order to emulate sequencing artifacts (see Methods).

The results on simulated 454 and Illumina reads are summarized in Fig. 5.2. `segemehl` performs best in both 454 simulations. In the data

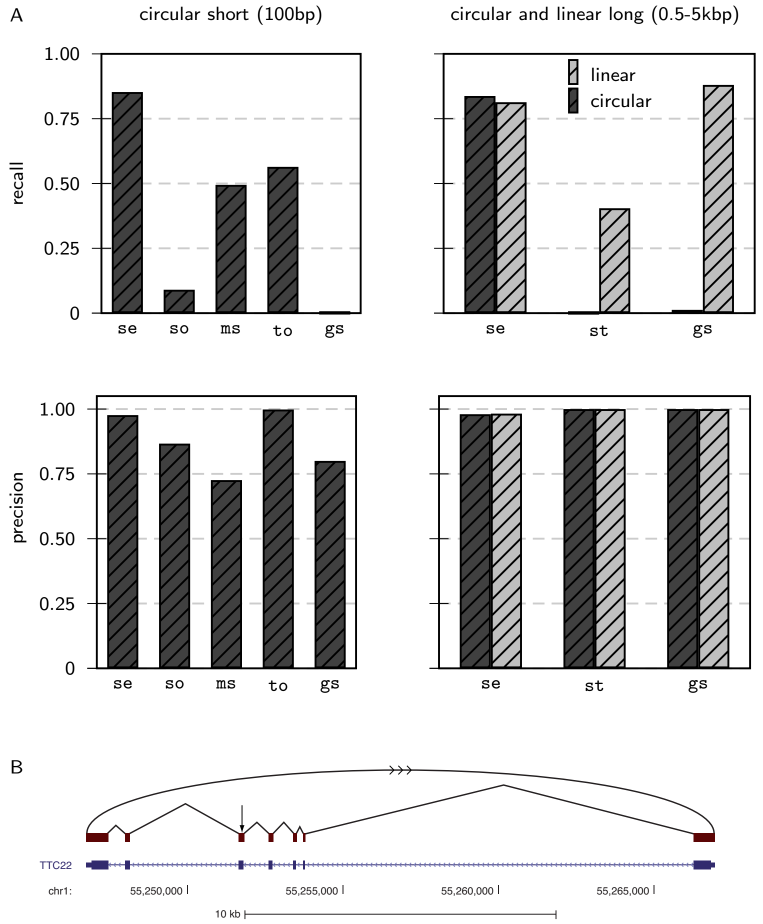


Figure 5.3: **Recall and precision of short circular, long circular, and long collinear transcripts.** For this benchmark, we have tested `segemehl`'s performance on sequence reads that were generated from the RefSeq database (A). To simulate sequencing errors, we applied an Illumina error model to the short circular reads (100bp) and a 454 error model to the long circular and collinear transcripts (0.5-5kB). On short circular transcripts, `segemehl` achieved a recall of more than 85%, outcompeting all other tools while maintaining a high precision of 98%. Using RefSeq transcripts of length 0.5-5kB, `segemehl` achieved a recall of more than 80% on circular and linear transcripts. Among the tools that were able to handle such long transcripts, `segemehl` was the only tool that was able to detect the circularization. For long collinear transcripts, `GSNAP` is slightly ahead of `segemehl` by 6% – at the expense of a nearly 2-fold increase in runtime (Tab. 5.1). (B) The RefSeq TTC22 transcript is an example for the simulated circularization. The arrow indicates the position, where the transcript has been artificially circularized. `SpliceMap`, `RUM` and `STAR` did not report any circular junctions (not shown). `STAR` and `GSNAP` were the only tools able to handle long reads. (`to: TopHat2; se: segemehl; ms: MapSplice; so: SoapSplice; st: STAR; gs: GSNAP`)

set with regular splice junctions, `segemehl` consistently recovers more than 90% of all simulated splice junctions. Its closest competitors, `GSNAP`, achieves recall between 81% and 92%. `STAR` is third, with less than 87% of recalled splice junctions. Probably due to length restrictions, `TopHat2` did not report any results after more than 1 week and was terminated. For irregular splice events, the difference is even more striking: while `segemehl` recovers approximately 90% of all simulated splice junctions, the next best competitor, `STAR`, achieves a recall of approximately 55%.

The improved performance on 454 reads does not significantly impair `segemehl`'s performance on Illumina data. All tools, including `segemehl` with a recall at the 95% level, report more than 90% of all simulated splice junctions. `RUM`, gaining an advantage by simultaneously aligning the reads to genome and transcriptome, performs best in this scenario. The best genome-only aligner is `STAR` with a recall of 98%.

As soon as trans-splicing events are involved, five of the seven alternative tools recover less than 80% of all splice junctions. `TopHat2` has a better sensitivity and reports more than 90% of the junctions. However, it is important to notice that its false positive rate of more than 10% is quite high. In contrast, `segemehl` identifies about 95% of all junctions in this data set and reports only 2% false positives. In the case of Illumina reads, the runtime of `segemehl` is comparable to most of the tools tested. Only `MapSplice`, `GSNAP`, and `STAR` were significantly faster than `segemehl` (Tab. 5.1).

In addition to these benchmarks that were all carried out with Illumina and 454 error models, we wanted to further investigate `segemehl`'s behaviour for reads with higher error rates, for example caused by multiple single nucleotide variations or –more importantly– less successful sequencing runs. Therefore, we carried out further benchmarks with higher error rates (up to 5% mismatches and indels) and varying coverages. As expected, the recall increases with higher coverages and drops with higher error rates. However, `segemehl`'s specificity is kept consistently at a very high level for all types of splice junctions (Fig. 5.4).

The performance on artifical short circular (100bp), long circular and long linear reads (0.5-5kb) is summarized in Figure 5.3. For the short reads, `segemehl` achieves a recall of 85% of all circular junctions using uniquely mapping split reads. The closest competitor, `TopHat2`, achieves 55% of recall. `SpliceMap`, `RUM` and `STAR` did not report any circular junctions (Fig. 5.3A). In contrast, `STAR` and `GSNAP` were the only tools that were in principle able to handle long reads. For linear transcripts, `GSNAP` is slightly ahead of `segemehl` by 6%, whereas `STAR` was merely able to recall 40% of all junctions. No circular junctions were recovered by `STAR` or `GSNAP` (Fig. 5.3B).

	`segemehl`	`TopHat2`	`SpliceMap`	`MapSplice`	`SOAPsplice`	`STAR`	`RUM`	`GSNAP`
Illumina normal								
10x	1401.97	1908.83	1907.25	968.96	1935.35	25.37	1634.24	922.10
15x	2058.28	2630.04	2613.53	1357.54	2924.20	39.90	2384.61	1381.43
20x	2709.30	3356.13	3226.79	1766.17	3888.79	57.73	2872.59	1842.11
Illumina trans								
10x	1466.49	1927.86	1983.13	1235.39	2249.18	26.32	1694.79	1436.51
15x	2122.52	2633.77	2602.47	1745.07	3360.47	41.26	2295.42	2123.29
20x	2790.97	3390.95	3293.05	2242.04	4475.16	61.26	3056.67	2819.13
454 normal								
10x	2446.53	-	2627.13	937.89	5397.87	35.61	2105.31	1651.65
15x	3602.95	-	3642.76	1311.12	8088.81	53.64	2963.79	2524.77
20x	4882.83	-	4633.10	1688.17	10796.57	72.89	3923.66	3346.16
454 trans								
10x	2548.14	-	2709.37	961.07	5629.92	33.50	2032.09	1983.12
15x	3764.13	-	3765.43	1357.22	8481.81	51.42	3023.82	2969.96
20x	5016.22	-	4825.93	1746.85	11232.61	69.46	3861.59	3926.48
RefSeq circular short								
-	238.38	749.96	-	546.60	831.52	-	-	1359.54
RefSeq circular long								
-	9643.94	-	-	-	-	486.64	-	16611.17
RefSeq linear long								
-	9849.77	-	-	-	-	1668.61	-	17512.11

Table 5.1: **User times for simulated reads for seven different split read aligners.** In the case of Illumina reads the runtime of `segemehl` is comparable to most of the split read aligners tested here. `STAR` has the fastest runtime. `TopHat2` did not finish after 1 week with long 454-like reads.

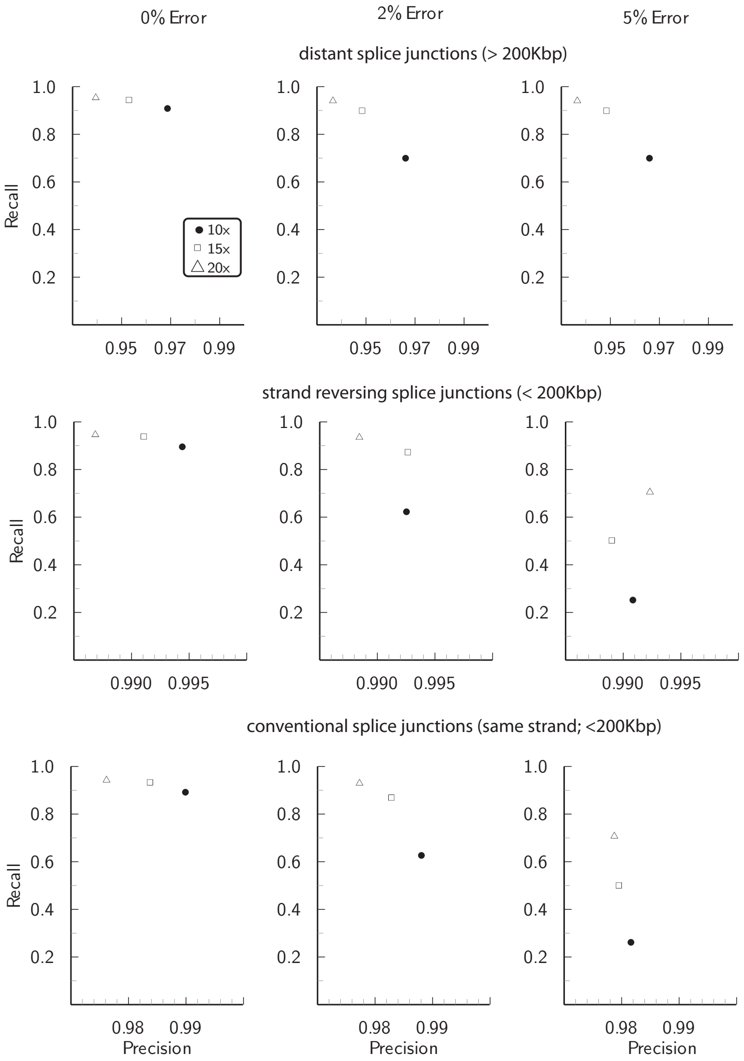


Figure 5.4: **Effect of sequence errors and coverage on `segemehl` alignments in terms of recall and precision.** Recall and precision were measured separately for long distant splice junctions (>200kb; top row), strand-reversing (middle row) and regular splice junctions (bottom row). In the graphs, coverage values (10-fold, 15-fold, 20-fold) are indicated as symbols. For each of these simulated splice junctions we simulated reads without errors (no mismatches and indels; left column), 2% error rate (middle column) and 5% error rate (right column). As expected, the recall increases with the coverage. Overall, segemehl shows a high precision. Interestingly, only the recall but not the precision declines, when errors are introduced.

Real data

We applied `segemehl` to a number of real data sets. Publicly available Illumina RNA-seq data sets of *D. melanogaster* [SRA:SRR166809] and a human melanoma sample [SRA:SRR018261-62] were downloaded from the NCBI short read archive. The 454 sequencing data for a human umbilical vein RNA-seq experiment [Djebali 2012] and RNA capture sequencing experiments of human fibroblasts [Mercer 2012] was retrieved from GEO under [GEO:GSM951482] and [GEO:GSE29040]. The RNA-seq sample for HEK293 cells investigated in [Memczak 2013] was retrieved under [GEO:GSE43574]. Finally, we have applied our algorithm to a *C. elegans* data set [SRA:SRX151602] for the investigation of RNA leader trans-splicing. All of these data sets are non-strand specific.

Split-mapping of an *Drosophila melanogaster* RNA-Seq data set resulted in the successful recovery of the previously described trans-splicing of MODMDG4 gene (Fig. 5.5 a) [Dorn 2001]. Most notably, `segemehl` revealed three alternative strand-reversing junctions consistent with a splice event between a common exon on the reverse strand to three exons encoded on the forward strand.

In a human melanoma transcriptome data set, the method identified the recently described CDK2-RAB5B read-through transcripts on chromosome 12 (cf. [Berger 2010]) (Fig. 5.5 b). In addition, `segemehl` identified a huge number of strand-reversing split reads located at the locus of the premelanosome protein (PMEL; SILV). This gene located on chromosome 12 encodes a melanocyte-specific transmembrane glycoprotein and is expressed under physiological conditions in melanosomes. It plays an essential role in the structural organization of premelansomes and has been suggested as a potential serum marker for melanoma (Fig. 5.5 c). More than 20% of the trans-splicing events detected in the sample occurred at this locus, making general errors in the RNA preparation and analysis highly unlikely. Thus, the PMEL locus might be an interesting target for further investigations.

We additionally applied our algorithm to a 454 data set published by [Mercer 2012] generated with RNA from human foot fibroblasts. As an example demonstrating the efficiency of `segemehl`, we consider the tumor suppressor gene p53 in (Fig. 5.6) This key regulator of the cell cycle is one of the most intensely studied genes because of its importance in cancer research. The functionally distinct variants and isoforms of p53 have been the focus of an intense research effort [Marcel 2011, Camus 2012]. Despite the attention this gene has already received, a re-analysis of the raw data [Mercer 2012] identified 3 previously unrecognized, canonical splice variants. We have validated all three novel isoforms in venous fibroblasts by PCR and sequencing (see below). Since we failed to validate the p53 isoforms in HUVEC cells (data not shown), these splice variants might be tissue-specific.

Novel transcripts were also predicted in a HUVEC 454 RNA-seq data set [Djebali 2012]. The example in (Fig. 5.6 B) shows two isoforms whose start is located anti-sense within the intronless TACSTD2 gene. Their first intron contains on the opposite strand the entire MYSM1 gene, a histone H2A deubiquitinase.

In a human prostate carcinoma cell line, we identified a transcript that aligns to adjacent regions on plus and minus strand of the genome. We validated the occurrence of this split using RT-PCR followed by cloning and sanger sequencing. Interestingly, the split is located in the 3′ UTR of the stearoyl-CoA desaturase gene SCD (see below), which has been implicated in prostate cancer [Kim 2011b].

For the RNA-seq data from HEK293 cells analyzed specificially for circRNAs in [Memczak 2013], we were able to recover all circRNAs that were experimentally validated by the authors of the original study (cf. [Hoffmann 2014a]).

To benchmark the speed of our split-read aligner, we have aligned four different data sets with `segemehl`, `STAR`, `SOAPsplice`, `GSNAP`, `TopHat2`, and `RUM`. With the exception of `STAR`, in most scenarios `segemehl` is faster or at par with the other tools tested (Table 5.3). Because of the usage of a full enhanced suffix array, the memory consumption of segemehl depends on the size of the reference genome (cf. [Hoffmann 2009]). Thus, it has the highest memory consumption among all tools tested. For large mammalian genomes `segemehl` may not be feasibly applied on computers with less than 50 GB of memory. The size of the read library, however, does not affect the memory consumption. Note that for smaller genomes the memory consumption is considerably smaller, e.g *C. elegans* 1.5 GB, *Drosophila melanogaster* 2.6 GB, and *Arabidopsis thaliana* 1.8 GB.

Spliced leader trans-splicing in *C. elegans*

	splice juncts.	trans split reads	% of trans split reads
known SL	6904	120998	68.1%
rRNA cluster	5358	38210	21.5%
RNA genes	207	1414	0.8%
others	2166	17016	9.6%

Table 5.2: **Trans-splice junctions involving SL and rRNA loci.** The majority of all split reads spanning more than 200K or split up between different chromosomes are aligned to known spliced leader positions or rRNAs. The rRNA cluster on chrI accumulates 21.5% of the trans-split reads, while the rest of RNA genes, including rRNAs outside the cluster on chrI, account for less than one percent of trans split reads.

Finally, we have tested our algorithm on the transcriptome of the nematode *Caenorhabditis elegans*. The roundworm is known to have an extensive amount of trans-spliced transcripts. Particularily, spliced leader sequences (SL) of about 22nt are spliced to the trimmed 5′-ends of many mRNAs [Blumenthal 2005]. The spliced leader sequences are encoded as part of small ncRNAs, the SL RNAs, in 28 loci scattered throughout the genome [Stricklin 2005, Ross 1995]. In order to test whether SL trans-splicing can be detected directly from the `segemehl` mapping results we re-analyzed a small part of the publicly available sequencing data generated by [Hillier 2009]. The *C. elegans* transcriptome data (SRX15106) was mapped using an increased accuracy of 95% (-A 95) and increased sensitivity for small splits (-Z 19) to the *C. elegans* reference genome ce6. Subsequently, we selected all split reads spanning more than 200K or split up between different chromosomes. All trans-splicing junctions reported by the `segemehl` mapping were required to have minimum split read support of 3. After masking rRNAs, all split-reads starting or ending in the rRNA cluster on chrI:15060287-15072132 we obtained approximately 9000 junctions linking loci with a distance of more than 200kb or on different chromosomes. These were supported by about 139000 split reads. More than 90% of them connect to the genomic coordinates of 3' end of one of the annotated the spliced leader sequence (Tab. 5.2). This simple survey accurately reproduces results from a recent detailed analysis of *C. elegans* trans-splicing [Allen 2011]. In particular we find that 70% of the ce6 mRNAs annotated in the UCSC genome browser are transspliced. We also recover the expected distribution of spliced leader usage: SL1 is by far the most frequently used leader sequence (85.9% of all SL junctions) followed by SL2 (13%), SLf (0.4%), SLb (0.3%), SLc (0.2%), SLd (0.2%), SLa (<0.1%). Other trans-splice junctions found during this excercise are subject to further research.

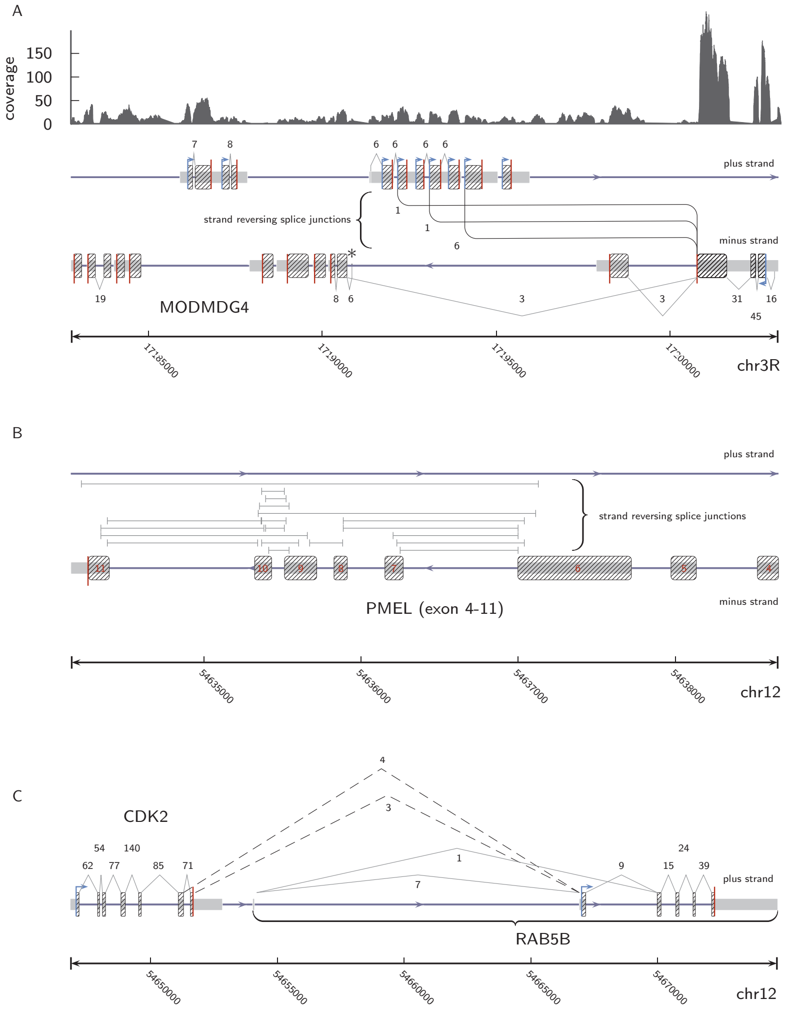


Figure 5.5: **Examples of (re-)discovered splicing events from single-end split reads.** (A) In *Drosophila melanogaster* `segemehl` recovered three different previously described trans-splicing junctions linking the minus encoded exon three as a donor in the MODMDG4 gene on chromosome 3R. The strand-reversing splice junctions are annotated between the plus- and minus strand. Note, that the direction of strand-reversing splice junctions, i.e. form the minus to the plus strand, was inferred from annotation and prior knowledge. This was necessary because the RNAseq library used was not strand-specific. (B) In the human melanoma transcriptome data set, `segemehl` identified a massive number of strand reversing splice junctions in the premelanosoma protein gene locus (PMEL). The split reads that support these junctions split from the plus strand to the minus strand and vice versa. Since we lack additional information, a direction of these junctions cannot be given. Note that only a selection of strand-reversing PMEL junctions is shown here. (C) In the same data set, `segemehl` reported two alternative transcripts linking CDK2 and RAB5B encoded on human chromosome 12. These junctions (dashed lines) are supported by split reads whose fragments map to the same strand, i.e. split reads that were not strand-reversing. Since the junctions exaclty hit annotated the boders of CDK2 and RAB5B exons we assigned them to the minus strand.

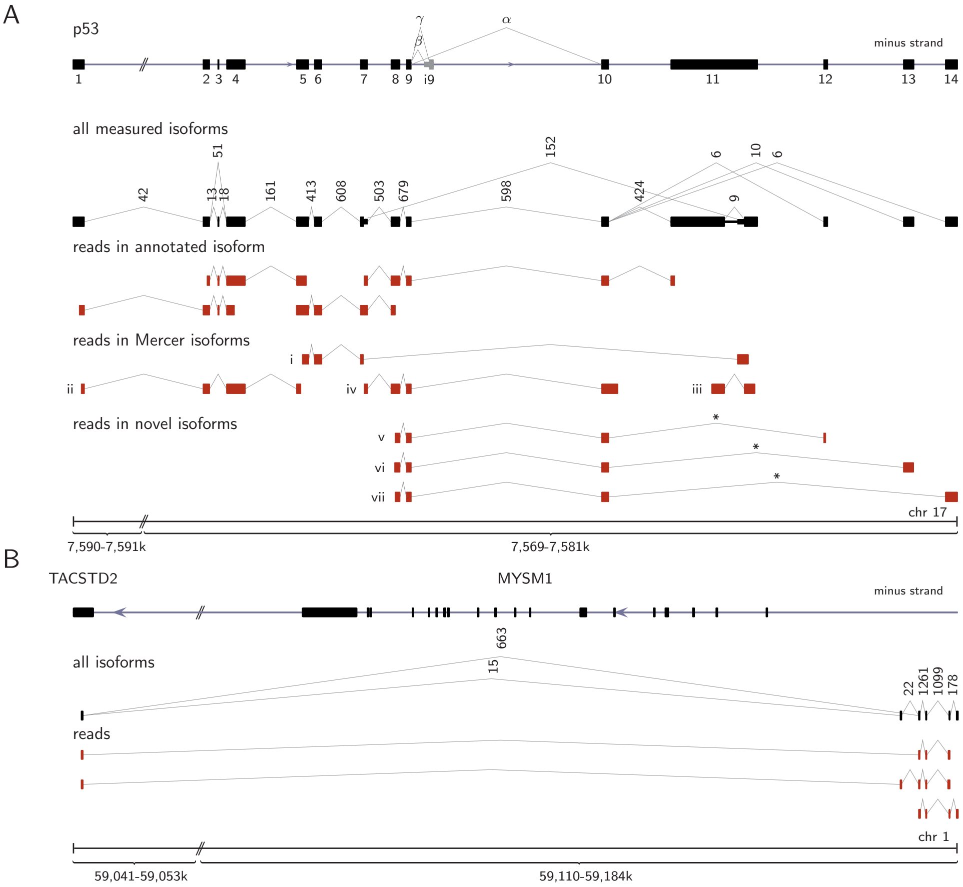


Figure 5.6: **Novel and known spliced transcript isoforms identified with long single-end 454 RNA-seq split reads.** (A) Transcript isoforms of the p53 gene. In addition to previously reported isoforms (i-iv) [Mercer 2012], we identified three novel canonically spliced isoforms (v-vii). Consistent with [Mercer 2012], β and γ isoforms are not expressed here. Each splice junction is labeled with its read support, i.e. the number of reads that map accross this junction. For better comparability with [Mercer 2012] the p53 gene, encoded on the minus strand of chr17, is shown in the direction of transcription from left to right. The junctions marked with an asterisk have been experimentally validated. (B) Unannotated transcripts in vicinity to TACSTD2 and MYSM1 gene recovered from a HUVEC RNA data set [Djebali 2012]. `segemehl` revealed the exon structure of two novel transcript isoforms comprising at least four exons. One exon common to both isoforms is mapped to the TACSTD2 gene. The associated introns enclose the MYSM1 gene locus. The putative gene structure is supported by three exemplary multi-split reads (not strand-reversing). Some of the splice junctions have already been reported by ENCODE/CSHL (HUVEC polyA+ RNASeq). The strandedness of the isoforms can not be inferred.

Validation of newly identified p53 isoforms

For the validation of the newly identified canonical splice junctions in p53 we used RNA from venous fibroblasts. The RNA was isolated with TRIzol reagent (Life Technologies) and treated with RNase-free DNase

(Qiagen) according to the manufacturers' instructions. The RNA samples were prepared in the context of another study recently published in PLoS Genetics [Holdt 2013]. The protocols for reverse transcription into cDNA are described therein.

PCR reactions were prepared in a final volume of 25 µl using AmpliTaq Gold(R) 360 DNA Polymerase (Life Technologies) and primers were selected to span two exons in order to avoid co-amplification of genomic DNA: common forward primer (5′-CCGAGAGCTGAATGAGGCCTTG-3′, 300nM) and isoform-specific reverse primers (isoform v 5′-CATCACACTGCATACCTTGAATGTATGC-3′; isoform vi 5′-CAGGCTAGAGTGCAATGGCGC-3′; isoform vii 5′-GGCTCACGCCTGTAATCCCAGTAC-3′; 300nM each). Expected PCR product sizes were 322bp, 213bp, and 178bp for isoform v-vii, respectively. Cycling conditions were 95°C for 10 minutes and 40 three-step cycles of 95° for 20 seconds, 60°C (isoform vi) or 62°C (isoform v/vii) for 30 seconds, and 72°C for 30 seconds. PCR products were subcloned using the TOPO TA Cloning Kit (Life Technologies) and sequencing reactions were performed with forward and reverse M13 primers (5µM, Life Technologies) and BigDye(R) Terminator v 3.1 chemistry (Life Technologies) according to the manufacturer's instructions an Applied Biosystems 3730xl DNA Analyzer. For details see supplementary material of [Hoffmann 2014a].

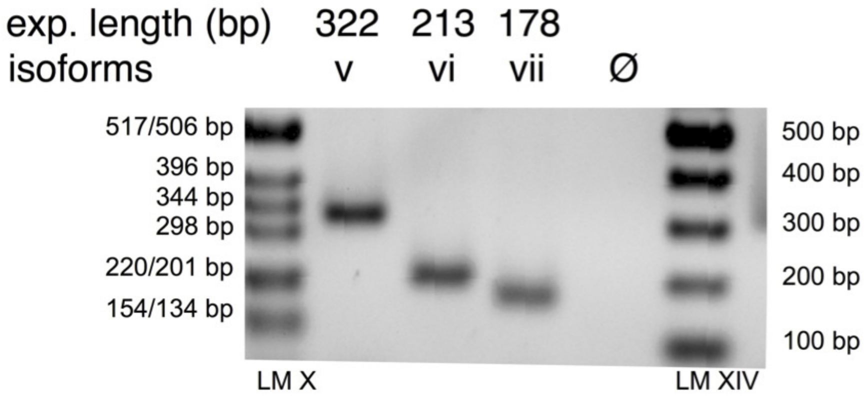


Figure 5.7: **PCR validation of three p53 novel isoforms on 2% LMP agarose gel.** Observed product sizes of PCR products were as expected. (LM X: Roche DNA Molecular Weight Marker X (0.07- 12.2 kbp); LM XIV: Roche DNA Molecular Weight Marker XIV (100-1500bp))

Sequencing and Validation of LnCaP trans splicing

Cells from the human prostate carcinoma cell line LNCaP were maintained and total RNA was isolated as described previously [Boll 2012]. Ribsomal RNAs were removed from total RNA using the Ribo-Zero Kit (Epicentre, Madison, WI) according to the manufacturer's instructions.

A strand-specific library for transcriptome sequencing was prepared using the ScripSeq Kit (Epicentre) following the manufacturer's instructions. The library concentration was determined using an Agilent 2100 Bioanalyzer system with a High Sensitivity DNA Kit (Agilent, Sanat Clara, CA) according to the manufacturer's instructions and relying on the concentration of fragments between 150 and 600 nt in size. 12 pmol of library were clustered on one lane of an Illumina paired-end flow cell and 2x100nt were sequenced according to the manufacturer's instructions using the v3 Cluster Generation and SBS Sequencing Kits (Illumina, San Diego, CA) on a HiSeq2000 system. Reverse transcriptase polymerase chain reaction (RT-PCR) was used to validate the observed split in total RNA of the human carcinoma cell line LNCaP.

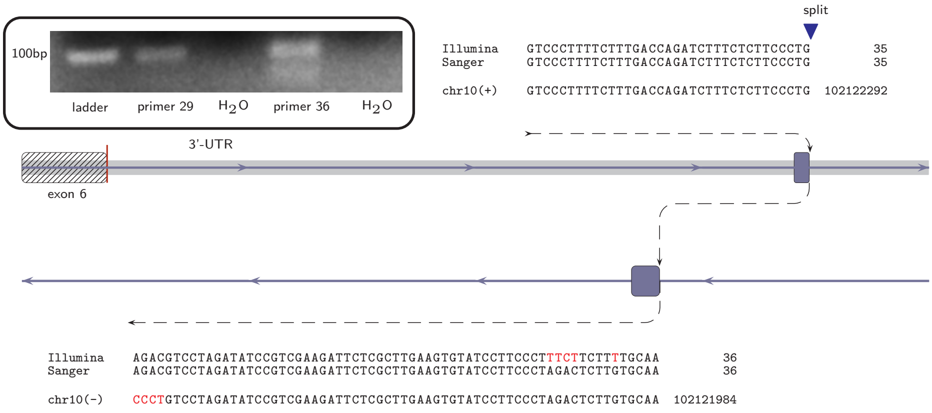
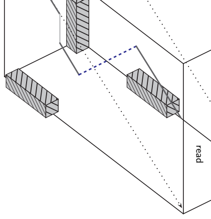

100bp
ladder
primer 29
H2O
primer 36
H2O
split
Illumina GTCCCTTTTCTTTGACCAGATCTTTCTCTTCCCTG 35
Sanger GTCCCTTTTCTTTGACCAGATCTTTCTCTTCCCTG 35
chr10(+) GTCCCTTTTCTTTGACCAGATCTTTCTCTTCCCTG 102122292
3'-UTR
exon 6
Illumina AGACGTCCTAGATATCCGTCGAAGATTCTCGCTTGAAGTGTATCCTTCCCTTTCTTCTTTTGCAA 36
Sanger AGACGTCCTAGATATCCGTCGAAGATTCTCGCTTGAAGTGTATCCTTCCCTAGACTCTTGTGCAA 36
chr10(-) CCCTGTCCTAGATATCCGTCGAAGATTCTCGCTTGAAGTGTATCCTTCCCTAGACTCTTGTGCAA 102121984

Figure 5.8: **Validation of a split observed in the prostate carcinoma cell line LNCaP.** The split aligns in parts to the forward and reverse strand of chromsome 10, respectively, and is located in the 3′ UTR of stearoyl-CoA desaturase (SCD). This enzyme is known to promote prostate cancer cell proliferation and transactivation of the androgen receptor, the key signaling pathway in this disease [Kim 2011b]. LNCaP total RNA was sequenced strand-specific using Illumina paired-end sequencing subsequent to ribosomal RNA removal. For validation, PCR primers were designed using the sequence reads exhibiting the split as a template. Conventional RT-PCR was performed using these primers and the amplicon of the expected size shown in the inset was cloned. Sanger sequencing of this clone confirmed the observed split.

	user time [hh:mm:ss]			
	SRR166809 [Graveley 2010]	SRR018261 [Berger 2010]	SRR018262 [Berger 2010]	SRR515313 [Djebali 2012]
`segemehl`	45:39:16	20:10:46	19:54:20	03:48:59
`STAR`	17:35:24	01:10:17	01:09:31	00:02:53
`SOAPsplice`	73:01:41	22:48:00	24:16:30	-
`GSNAP`	68:18:41	17:38:13	23:01:29	06:45:17
`RUM`	*	18:42:19	17:04:37	20:35:25
`TopHat2`	69:59:37	28:16:11**	28:58:29**	-
sequencer:	illumina	illumina	illumina	454
species:	drosophila	human	human	human
type:	paired end 100bp	paired end 50bp	paired end 50bp	single end
size:	4.6GB	1.6GB	1.6GB	197MB

Table 5.3: **Runtime comparision of read aligners with different real life data sets.** `SOAPsplice` and `TopHat2` did not produce any output on 454 data sets. (*`RUM` has chrUextra in genome (output >200GB), **run with default parameters killed after 4 days; restarted with –no-coverage-search)



Chapter 6

A data adaptive model for variation detection

Contents

6.1 Introduction 89
6.2 The statistical model 92
6.3 Results 96

Several sources of noise obfuscate the identification of single nucleotide variation in next generation sequencing data. Not only errors introduced during library construction and sequencing steps but also the quality of the reference genome and the algorithms used for the alignment of the reads play an influential role. It is not trivial to estimate the influence these factors for individual sequencing experiments. We introduce a simple data-adaptive model for variant calling. Several characteristics are sampled from sites with low mismatch rates and uses to estimate empirical log-likelihoods. These likelihoods are then combined to a score that typically gives rise to a mixture distribution. From these we determine a decision threshold to separate potentially variant sites from the noisy background. In simulations we show that the proposed model is at par with frequently used SNV calling algorithms in terms of sensitivity and specificity. The application to next-generation sequencing data reveals stark differences of the score distributions indicating a strong influence of data specific sources of noise. The proposed model is specifically designed to adjust to these differences. This chapter is based on [Hoffmann 2014b].

6.1 Introduction

Recent studies report a strikingly low concordance of currently available methods and pipelines for SNV calling, both somatic and germline. The results indicate that both the computational methods as well as the sequencing protocols have major impact on the sensitivity and specificity of variation calling tool [O'Rawe 2013]. Specifically, the allelic fraction as well as the coverage of the variant allele are major determinants for the statistical benchmarks [Xu 2014, Yu 2013]. Practical guidelines of SNV callers

protocol step	error source	effect
library	repetitive DNA/RNA	multiple hits, high or low coverage[1]
	RNA reverse transcription	erroneous cDNA
	low library complexity	non-uniform coverage
	contamination	false mapping, low coverage
	PCR-based errors	erroneous reads, false mapping
	amplification bias	non-uniform coverage, duplicates
sequencing	sequencing bias	non-uniform coverage
	sequencing errors	erroneous reads, false mapping
alignment	erroneous reference sequence	pseudo-erroneous reads, false mapping
	alignment heuristics	low coverage, false mapping

Table 6.1: Potential sources of error in the DNA and cDNA sequencing protocol steps.

such as GATK [McKenna 2010] or SAMtools [Li 2009b] suggest to apply rigorous postprocessing filters to reduce the number of false positive calls. Other studies indicate that the application of these filters lead to a substantial improvement of the concordance of the callers [Liu 2013]. Nevertheless, applying stringent thresholds for variables such as the strand bias, the coverage or read start variation bears the risk of losing important information [Pabinger 2013]. These authors emphasize that the different algorithmical and statistical components of a variant caller have to be evaluated as a whole and cannot not be meaningfully judged as single components.

Obviously, if DNA library preparation protocols and sequencing machines were able to produce error-free and unbiased sequences of sufficient length the task of variant calling would be easy. Due to various error sources and technical limitations of library preparation, sequencing, and alignment, however, a substantial level of noise complicates the analysis.

Since these factors can not be totally controlled during the experiment it seems reasonable to adjust the thresholds for calling a variant depending on the separability of noise and signal, i.e., the true variants. During amplification incorrect nucleotides are incorporated with some error rate ε_f and during the sequencing step incorrect nucleotides are called with the rate ε_g. After the alignment of the reads to a reference sequence we may observe these errors as mismatches or indels. Additional mismatches and indels are caused by these reference and alignment errors (ε_a). Some of the factors influencing the error rates are listed in Table 6.1.

The mismatch rate of a genomic site can be assumed to be the sum $\delta = \varepsilon + \beta$, where β represents the biological variation and ε is the compound effect of the technical errors ε_f, ε_g, and ε_a. Fig. 6.1 summarizes this situation.

The two most commonly used tools for SNV calling methods, SAMtools and GATK, employ probabilistic models for variant calling. Specifically, the algorithm used by SAMtools [Li 2011] is based on the likelihood of a genotype which is computed as

$$L(g) = \frac{1}{m^k} \prod_{j=1}^{l} [(m-g)\zeta_j + g(1-\zeta_j)] \prod_{j=l+1}^{k} [(m-g)(1-\zeta_j) + g\zeta_j], \quad (6.1)$$

where g denotes the number of reference alleles, m the ploidy, k the number of reads seen at a site, and ζ the error probability delivered by the sequencer. Equation 6.1 assumes that the first l bases are identical to the reference, the subsequent bases are not. Subsequently, from this a likelihood for the allele count $L(c)$ is obtained. Using the observed allele frequency spectrum ϕ_k as prior information a posterior probability

$$\frac{\phi_k L(c)}{\sum_l \phi_l L(l)} \quad (6.2)$$

is computed, and a variant is called if it exceeds a certain specified threshold.

The GATK pipeline uses a related probabilistic model for calling variants [DePristo 2011]. Similar to SAMtools, the probability $\Pr\{D_j|A\}$ of observing the base D_j under the hypothesis that A is the true base is calculated by

$$\Pr\{D_j|A\} = \begin{cases} 1-\zeta_j & D_j = A \\ \zeta_j \ \Pr\{A \text{ is true}|D_j \text{ is miscalled}\} & \text{otherwise}, \end{cases} \quad (6.3)$$

where $\Pr\{A \text{ is true}|D_j \text{ is miscalled}\}$ is a precomputed, sequencer specific lookup table.

Using prior information based on precomputed heterozygosity estimations GATK evaluates the posterior probabilities of a site to be variant. As with SAMtools calls are determined using fixed preselected thresholds.

Here, we propose a simple probabilistic model for variant calling using a data adaptive threshold on the scale of log-odd-ratios computed from empirical distributions of certain site characteristics. Our approach allows to optimally separate simulated SNVs from the noisy background without specification of a threshold for posterior probabilities. In brief, our model starts out by evaluating the mismatch frequencies δ in a data set. Subsequently, we sample several characteristics of the sites with small δ that serve as empirical null model. The fundamental idea used here is that the vast majority of sites is invariant and thus allows to capture the features of the data specific error model. These characteristics are then used to form empirical log-likelihoods that are combined to a log-odds type score. Typically, we observe a mixture distribution of two score populations, which we then separate by a decision threshold.

Next, we discuss the details of our model and the proposed data-

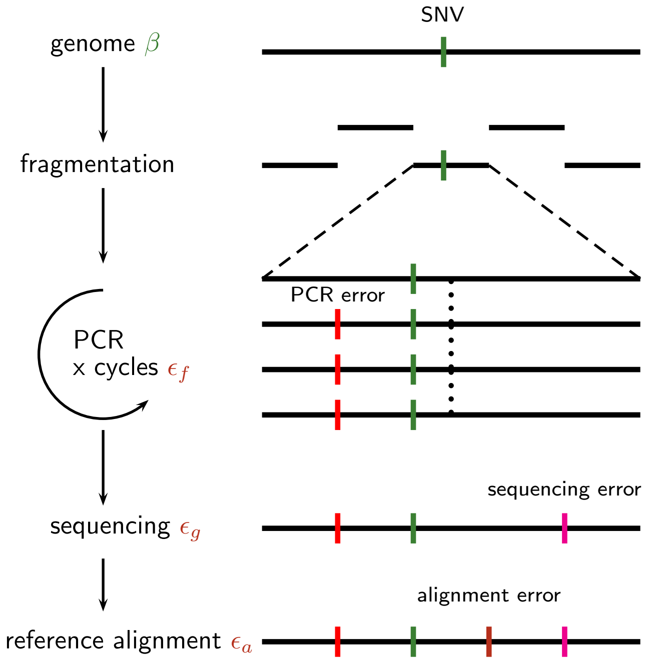


Figure 6.1: **Accumulation and sources of errors in next generation sequencing reads.** The identification of single nucleotide variations (green dashes) is complicated by various sources of error. PCR errors accumulate during the amplification and sequencing step. After fragmentation single fragments undergo several amplification cycles. Errors are introduced with a rate δ_f (red dashes). Further errors are accumulated during sequencing (δ_g). During the alignment to the reference further mismatches and indels are introduced (δ_a).

adaptive variant calling algorithms. Subsequently, we apply our approach to both synthetic and next generation sequencing data from various species. An reference implementation in C99 of our method is available at `http://www.bioinf.uni-leipzig.de/Software/`.

6.2 The statistical model

Notation

We denote the position of a site in the reference genome by $i \in [1, \ldots, n]$ where n is the genome length. After aligning the reads to a genome each reference base typically has several nucleotides aligned to it. We refer to the set of all aligned nucleotides as the *cross section* C^i at position i. The coverage at position i is the size $|C^i|$ of this set. We use the index $j \in [1, \ldots, |C^i|]$ to refer to a specific read. The length of read j aligned to site i is denoted by m_j^i, and the position of a nucleotide in a read is denoted by $k_j^i \in \{1, \ldots, m_j^i\}$. For simplicity, we occasionally leave out the index i when there is no danger of ambiguity.

The nucleotide in a cross section can be partitioned into sets of match (M) and mismatch ($\overline{M}$) nucleotides so that $C^i = M^i \dot{\cup} \overline{M}^i$. The variant calling algorithm described below uses the partition $\{M^i, \overline{M}^i\}$ at each position i to compute an overall score for this particular site.

Biological versus technical variation

The *mismatch rate* $\delta^i = |M^i|/|C^i|$ is the observed number of mismatches divided by the coverage. The mismatch rate $\delta^i = \beta^i + \varepsilon^i$ may be decomposed into biological and technical variation, where ε^i denotes the technical error that accumulated during the preparation, sequencing and alignment steps and β^i denotes the biological nucleotide variation at site i.

We aim to distinguish biologically variant positions ($\beta^i > 0$) from non-variant positions ($\beta^i = 0$), based on the observed mismatch rates δ^i and site characteristic scores derived from sequence data or produced during sequencing.

Cross sections with high mismatch rates are indicative of biological variation in the sample, whereas in cross sections with small mismatch rate the mismatches are more likely due to technical errors. Conversely, in the overwhelming majority of cross sections C^i we may assume that there is no biological variation present, i.e. $\beta^i = 0$, and thus mismatches are only caused by technical errors. For use in the variant calling score we estimate $\Pr(\delta_i)$ for by tabulating the respective frequencies across the genome. For computationally efficiency we only consider variant sites. To obtain the log-likelihoods of certain site characteristics we uniformly sample only from sites with $0 < \Pr(\delta_i) < 0.5$. The default value for the sample size in our current implementation is 100000.

The degree of biological variation depends on the type of genome. For heterozygous genomes one expects to find predominantly SNP alleles with $\beta^i = 0.5$ or $\beta^i = 1.0$, whereas cancer tissues may show mutations with $0 < \beta^i \leqslant 1$ depending on the heterogeneity and cancer cell content of the sample. Similarly, arbitrary values of β_i will appear in whole population sequencing data. Accordingly, we expect different values of β^i for mixtures of sequencing data from different individuals. The variant calling algorithm introduced in the following makes no assumptions concerning the presence of diploid genomes, knowledge about the ploidy, homo- or heterozygosity.

Site characteristics

In addition to the partitioning of nucleotides at a given site into match and mismatch sets, our algorithm uses the following information, which typically reported by the sequencer or the read mapper for every site i and read j:

- the nucleotide qualities (Q),
- relative read position (P),
- errors in the alignment (R), and
- the number of multiple hits (H).

The nucleotide qualities take on values between 0 and 1 and are given as $Q = 1 - \zeta$, i.e., as probability of a base being correct, with values close to 1 corresponding to optimally accurate sequencing. We directly use Q in computing the variant calling score.

The relative read position is given by $P = k_j^i / m_j^i$. For the construction of our variant calling score we employ the probability $\Pr(M|P)$ of a match at a given read position, along with the maximum $P_M = \max_P \Pr(M|P)$. The probability of a mismatch is then given by $\Pr(\overline{M}|P) = 1 - \Pr(M|P)$, and its maximum $P_{\overline{M}} = \max_P \Pr(\overline{M}|P)$. We estimate the probability $\Pr(M|P)$ empirically, i.e., by appropriately counting matches and mismatches over all sites and reads.

The errors in the alignment is an integer value greater or equal to zero, and denoted here by R. Finally, the number of multiple hits H describes the number of alignments for each read. The multiplicity of an alignment yields information on the repetitiveness of a genomic region. As above for the relative read position, we tabulate the occurrence of matches for each value of R and and H and correspondingly obtain estimates of the probabilities $\Pr(M|R)$ and $\Pr(M|H)$.

Scores for distinguishing variant and non-variant sites

Informally, in a non-variant cross section $(\beta^i = 0)$ we expect that the probability of a match base increases with high nucleotide qualities (good sequencing), low read error rates, few multiple hits and good read positions. Conversely, the probability of mismatching bases in non-variant cross sections increases with low nucleotide qualities (poor sequencing), high read error rates, multiple hits and error-prone read positions. For variant sites with $\beta > 0$ we expect to have high nucleotide qualities, good read positions and few multiple hits also for the mismatch bases. Consequently, for distinguishing variant from non-variant sites only the mismatching bases are relevant.

We introduce four log-odds ratios to formalize and summarize the evidence for a variant over a non-variant based on the above four site characteristics.

$$\Delta_Q = \frac{1}{|C|} \sum_{x \in \overline{M}} \log \frac{Q_x}{1 - Q_x}$$

for base qualities,

$$\Delta_P = \frac{1}{|C|} \sum_{x \in \overline{M}} \left(\log \frac{\Pr(M|P_x)}{1 - \Pr(M|P_x)} + \log \frac{P_{\overline{M}}}{P_M} \right)$$

for read positions, which are rescaled by their respective maximally attained values P_M and $P_{\overline{M}}$,

$$\Delta_R = \frac{1}{|C|} \sum_{x \in \overline{M}} \log \frac{\Pr(M|R_x)}{1 - \Pr(M|R_x)}$$

for read errors, and

$$\Delta_H = \frac{1}{|C|} \sum_{x \in \overline{M}} \log \frac{\Pr(M|H_x)}{1 - \Pr(M|H_x)}$$

for multiple matches. Only reads with mismatching bases in a cross-section are used for estimation. If there are only match bases in a cross-section , i.e. if $|\overline{M}| = 0$ then the scores are set to 0.

Variant calling with adaptive threshold

From these log-odds ration we now construct a total score for variant calling by computing, at any position i,

$$S_i = \Delta_{P_i} + \Delta_{Q_i} + \Delta_{R_i} + \Delta_{H_i} + \log \Pr(\delta_i)\,.$$

This score comprises the four summaries of the site characteristics, as well as the log-probability of the observed mismatch rate δ_i, i.e., the observed number of changes at position normalized by coverage. Low probabilities for δ_i strongly penalize the score.

For variant calling we now proceed as follows. We assume that the majority of the sites are non-variant, and only a smaller part is variant, with $S_i > 0$. Thus, the observed distribution of S_i will be a mixture distribution, consisting of a null distribution corresponding to the invariant sites and an alternative "contamination" component corresponding to the variant sites.

In order to find an optimal adaptive cut-off separating the background from potential variants we estimate the densities by fitting a natural spline using Poisson-regression to the histogram, following the procedure described by [Efron 1996]. Subsequently, we numerically find the location with the minimum density and use it as threshold for separating the two score populations. The corresponding threshold is denoted by S^*. In rare cases there is no minimum in the histogram; we then use the upper 95% quantile as threshold. Once the threshold S^* is established, we declare all

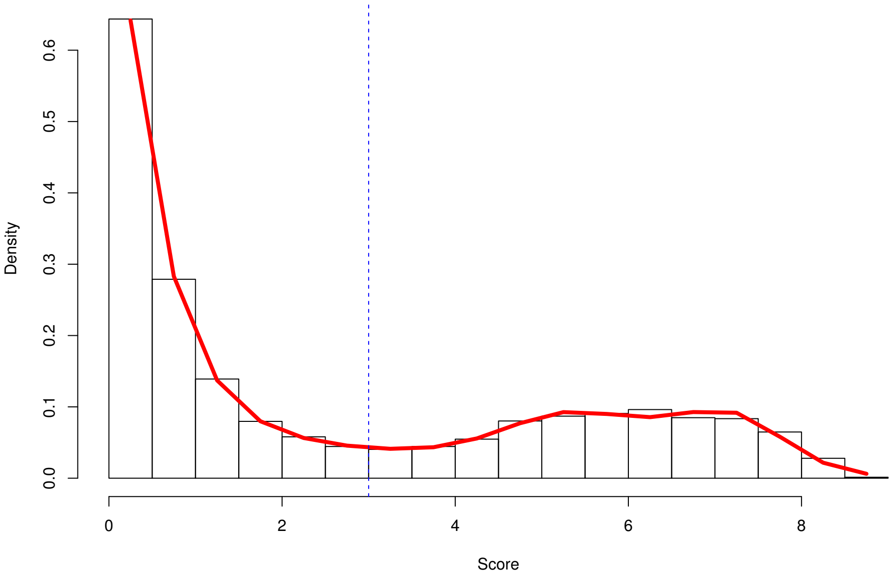


Figure 6.2: **Adaptive cutoff determination.** The figure shows the distribution of the empirical scores $S_i > 0$ (red continuous line) and the cutoff S^* (dotted blue vertical line). The threshold separates potential variants from background signal.

sites $S_i > S^*$ to be variant.

In Fig. 6.2 this procedure is illustrated using actual data from *A. thaliana*.

6.3 Results

Simulation study

To evaluate the behavior of our data adaptive model we compared it with the two frequently used SNV callers GATK [McKenna 2010] and SAMtools [Li 2009b]. The precise command line settings are compiled in the Appendix.

We simulated next generation sequencing data for the human chromosome 21 using GemSIM (version 1.6) [McElroy 2012] with the default model and coverages ranging from 10, 20, 30, 50, 100, to 200-fold. The simulated content of the variant allele was either 0.2 or 0.5. Simulated read length was 100. For mapping we used the aligners that are recommended for each method. Specifically, we used BWA [Li 2009a] to generate the alignments for GATK and SAMtools. For the reference implementation of our model we used `segemehl` [Hoffmann 2009]. After mapping and variant calling we collected for each combination of coverage and variant allele frequencies the number of false positives (FP), true positives (TP), false negatives (FN), and true negatives (TN). From this we computed the recall (sensitivity) $\mathsf{SENS} = \mathsf{TP}/(\mathsf{TP}+\mathsf{FN})$ and the positive predictive value $\mathsf{PPV} = \mathsf{TP}/(\mathsf{FP}+\mathsf{TP})$, a measure for the potential overcall of a method. For the proposed data adaptive model we investigated the score distribution for all 12 experiments (Fig. 6.3). Except for the combination of low coverage (10x) and low variant allele content (20%) we observe the presence of two populations. The separability of these score populations improves with increasing coverage and variant allele content. In each case, the minimum score for variant calls was automatically set to the value where the density of scores > 0 attains its first first local minimum. Subsequently all positions with a score equal or greater were called as SNV and compared to the other callers.

All of the tested programs show a good recall and positive predictive value in all 12 simulations. For low allele contents in conjunction with low coverages, however, SAMtools attains comparably low positive predictive values. Surprisingly, after reaching a maximum recall for the coverage of 100, the recall drops substantially for coverage 200. For the simulations with 50% allele content, all tools show high recalls and good positive predictive values. Again, SAMtools achieves only a comparably low positive predictive value for poorly covered SNVs (Fig. 6.4). Except for the lowest coverage, all tools performed well on these data sets.

Application to data sets

The good overlap between the different methods in our simulation study as well as the small number of false positives is in stark contrast to the experience of greatly differing variant calls in real life data (e.g. [O'Rawe 2013]). We therefore applied our model to diverse real data from both diploid and haploid organisms.

Paired end next generation sequencing data for *Arabidopsis thaliana* (SRR519713), *Escherichia coli* (ERR163894) and *Drosophila melanogaster* (SRR1177123) was downloaded under the respective accession numbers from the Short Read Archive (`www.ncbi.nlm.nih.gov/sra`). The *Arabidopsis* data was aligned to the to the reference genome version 10.5. With the data set SRR519713 we obtained a coverage of ~30-fold. The *E. coli* data set was aligned to the reference genome *E. coli k12* assembly v1.16. With ERR163894 we obtained a coverage of ~60-fold. Finally, SRR1177123 was aligned to the *D. melanogaster* reference version dm3. The coverage was ~25-fold. For the alignments, calling and filtering we used standard parameters. Precise settings are given in the Appendix.

The score distributions are shown in first line of Fig. 6.5. In the case of the plant *A. thaliana* and the procaryote *E. coli*, a clear separation of two populations is observable. On the other hand, the separation of the score

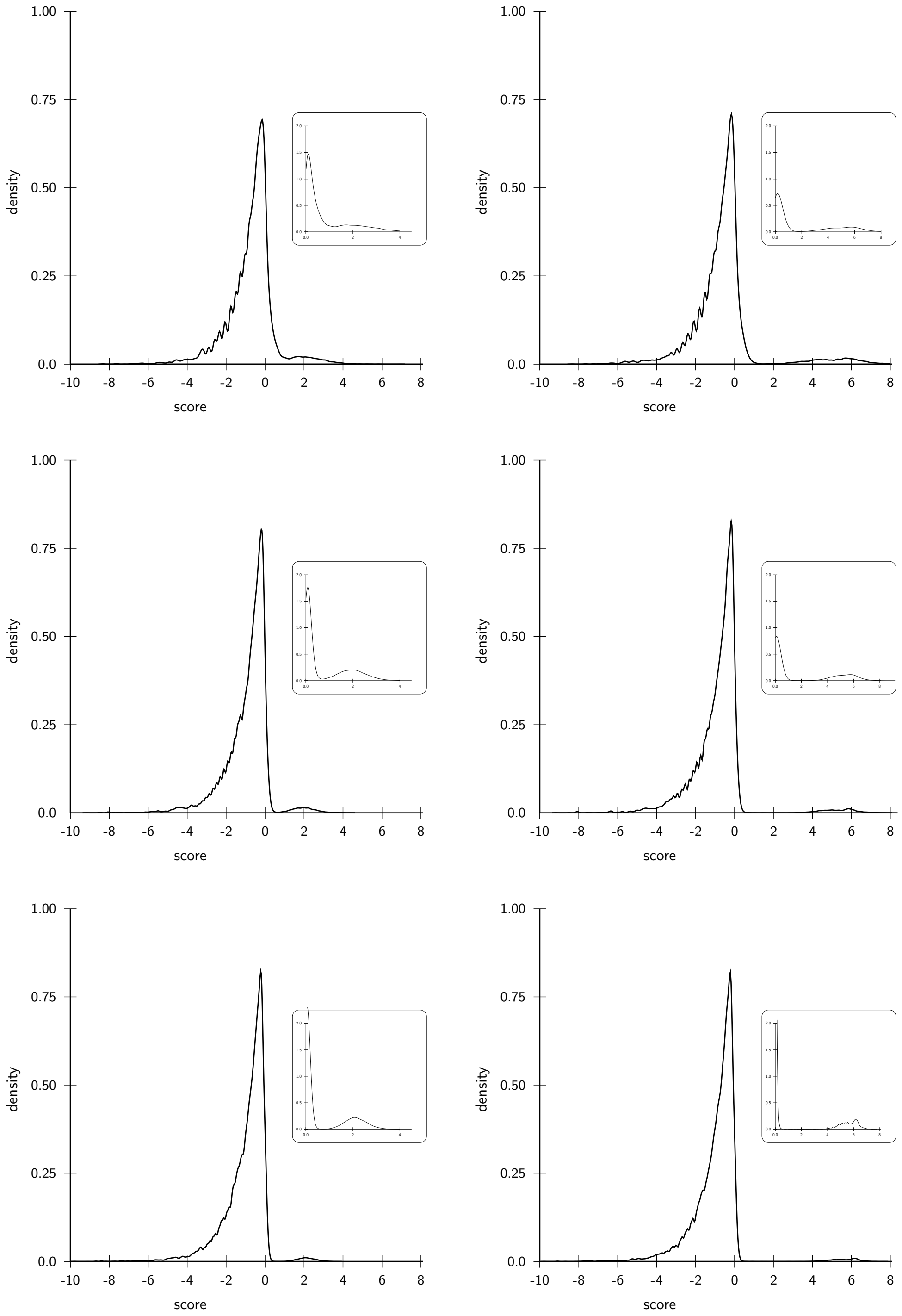


Figure 6.3: **Score distributions of simulated SNV data**. The left (right) column shows the score distributions of simulated SNVs with 20% (50%) variant allele content for 20-fold (top), 50-fold (middle) and 100-fold(bottom) coverage. The insets show the density of scores > 0. With the increase of coverage a population of scores > 0 is clearly distinguishable from the background.

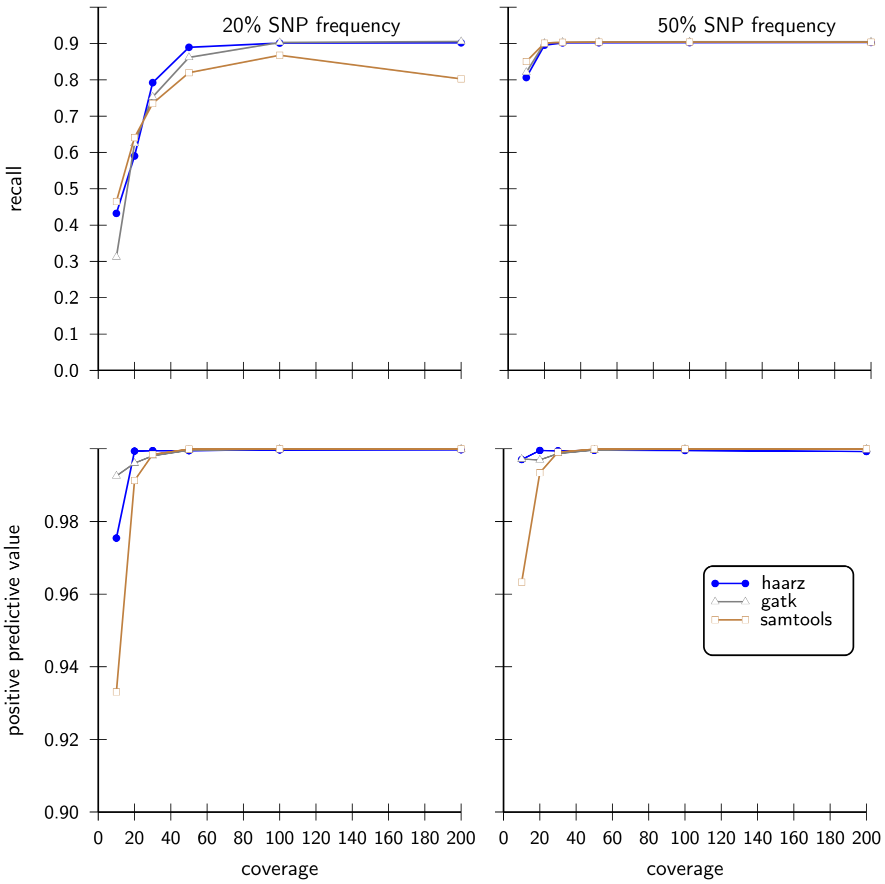


Figure 6.4: **Statistical performance measures on simulated data sets.** The data adaptive model was compared to SAMtools and GATK in terms of recall and positive predictive value. SNV calling was performed on twelve different data sets varying in the content of the variant allele (20% and 50%) as well as the simulated coverage (10-200). In all of these scenarios the data adaptive model is at par with both alternative callers.

populations in *D. melanogaster* data set is less pronounced.

In the lower part of Fig. 6.5 we show the concordance of variant calls of three investigated methods for the three data sets. For the well separable cases, the number of calls made by the data adaptive model is equal or higher as compared to the two competing callers. For the *D. melanogaster* data set our approach is more conservative and reports fewer variants. Most of these, however, are also found by SAMtools and GATK. About 92% percent of the calls from our model are also supported by both of the other callers and only 4% are not supported by any of the two alternative approaches. From the score distributions for *D. melanogaster* it is clear that there is a large overlap of the two score populations and hence the choice of S^* necessarily depends on the desired specificity and/or sensitivity.

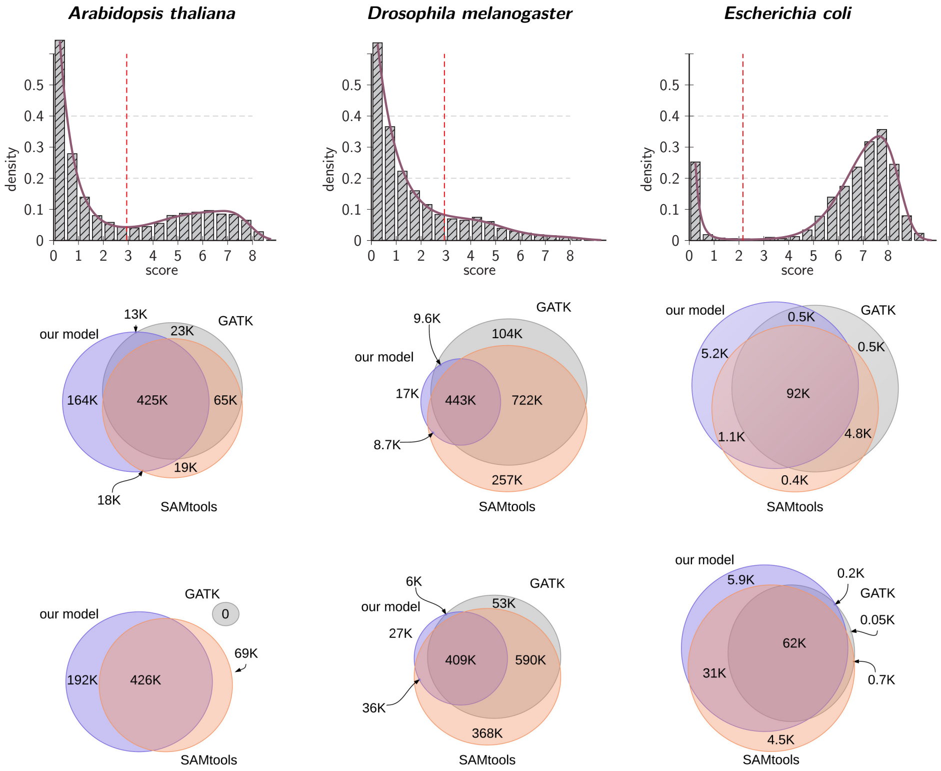
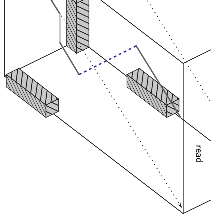


Figure 6.5: **Score distributions and congruence of variant calls in next generation sequencing data**. While a clear separation of score populations is observable for the diploid *A. thaliana* and the haploid *E. coli*, only a shallow minimum can be observed in the case of *D. melanogaster*. Hence, in the latter case our model automatically adjusts to the data such that the calling appears to be more conservative: less than 4 percent of the calls are not supported by any caller and 92% of its calls are supported both by GATK and SAMtools. On the other hand, our model calls more variants in the other two cases.



Chapter 7

Biological applications

Contents

7.1 **The primary transcriptome of *H. pylori*** **101**
  7.1.1 dRNAseq of the *H.pylori* transcriptome 102
  7.1.2 Mapping and basic sequence analysis 103
  7.1.3 Identifcation of start sites 105
  7.1.4 Differential expression analysis 109
7.2 **An insight to post-transcriptional modification** **111**
  7.2.1 Short RNA sequence data 111
  7.2.2 Chemical modifications 113
  7.2.3 Non-templated CCA-additions 115
7.3 **Mutated splice sites in Lymphoma samples** **117**
7.4 **Short read alignments to repetitive regions** **119**

In the context of this work several high throughput sequencing analysis projects have been carried out in a variety of species. The mapping and visualization of the data directly lead to the identification of new RNA species that escaped previous analysis and mapping methods due to their short length and high “error”-rates. A major milestone was the analysis of the primary transcriptome of the major human pathogen Helicobacter pylori. Furthermore, the devised algorithm was able to detect post-transcriptional modifications that escaped previous analysis and led to novel insights in this different layer of modification. This chapter summarizes the mapping results and basic sequence analysis steps from [Sharma 2010, Findeiss 2011]. The algorithms described above have been successfully applied to reveal new splice events caused by mutations in Lymphoma samples and recovered CHiP-Seq signals in repetitive genomic regions.

7.1 The primary transcriptome of *H. pylori*

The microaerophilic bacterium Helicobacter *pylori* is one of the major human pathogens that causes one of the most frequent infections of the world (cf. [Brown 2000]) with varying infection rates from 10% to 70%.

Helicobacter pylori infections are more frequently observed in developing countries [Bruce 2008]. It is one of the first bacterial agents known to cause cancer [Kusters 2006]. The nobel laureates Marshall and Morris demonstrated in the 80's that this bacterium is able to colonize the human stomach causing severe inflamations and eventually ulcerations as symptoms of the so-called peptic ulcer disease [Marshall 1985, Morris 1987]. Pathophysiologically this bacterium is of special interest since it colonizes and flourishes in a lowly oxygenized and highly acidic environment. To evade the gastric digestion it penetrates the gastric mucus and produces large amounts of ammonia to neutralize the acidity. The complete sequence of the circular Helicobacter pylori (strain 26695) genome was already published in 1997 by Tomb et al. [Tomb 1997]. Meanwhile, the genomic sequences of three more strains, namely HPAG1 [Oh 2006], J99 [Alm 1999], Shi470 (no publication), and G27 [Baltrus 2009] are available to date in public data bases. All genomes have a size of approximately 1.67Mbp and contain approximately 1500 predicted and confirmed genes. The predictions mostly rely on computational methods and need to be verified by experimental data. RNA sequencing is one way to confirm the predictions. One has to keep in mind though that predicted genes that can not be confirmed may just be lowly expressed or expressed under special conditions. Thus, the falsification of genes is much harder.

7.1.1 dRNAseq of the *H.pylori* transcriptome

In contrast to the general term transcriptome the *primary transcriptome* refers to those transcripts that have not been altered by post-transcriptional regulation. Thus for the analysis of the unaltered transcriptome a specialized experimental protocol had to be devised that specifically enriches immature RNA. The experimental protocol by the Vogel group of the Max-Planck-Institute for Infection Biology, the differential RNA sequencing approach (dRNA-seq), relatively increases the concentration of native transcripts by specifically degrading RNAs with a 5′-monophosphate using a terminator exonuclease (TEX) treatement. Primary transcripts including most coding messenger RNAs (mRNAs) and short non-coding RNAs (sRNAs) are protected from degradation by a 5′-triphosphate. This approach does not only allow to discover post-transcripitonal processes but allows to detect transcription start sites. To achieve this TEX-treated and untreated RNA libaries have to be prepared for sequencing. In this work, the Helicobacter pylori strain 26695 was exposed to a number of different conditions summarized in table 7.1.

Prior to library construction one fraction of the cells was treated with TEX, the other fraction remained untreated. TEX treated cells are denoted PL^+, AS^+, ML^+, HUH^+ and AGS^+, while the untreated cells are denoted PL^-, AS^-, ML^-, HUH^- and AGS^-. The libaries were constructed for the

Condition	Abbrev.	Description
mid-logarithmic phase	ML	grown on agar/blood media under microaerobic conditions
cell culture flask	PL	grown in cell culture flasks without agar/blood medium
actinomycin D	ML	grown on agar/blood media under microaerobic conditions treated with actinomycin
acid stress	AS	incubated at a pH of 5.3 for 30 minutes
co-culture with Huh7	Huh7	co-cultre of H. *pylori* w/ human hepatocarcinoma line
co-culture with AGS	AGS	co-cultre of H. *pylori* w/ human stomach adenocarcinoma line

Table 7.1: Growth conditions in the dRNAseq experiment for Helicobacter pylori. Mid-log cells were exposed ot acid stress and co cultured with human hepato- and stomach adenocarcinoma cell lines

454 pyrosequencing technology. A size fraction step to specifically enrich short RNAs was omitted. In brief, equal amounts of -/+ TEX treated RNA were poly(A)-tailed using poly(A) polymerase, followed by ligation of an RNA adapter to the 5'P RNA fragments. First-strand cDNA synthesis was performed using an oligo(dT)-adapter primer and M-MLV-RNaseH-reverse transcriptase. Incubation temperatures were 42C for 20 min, ramp to 55C, followed by 55C for 5 min. A total of 3.7 million cDNAs were sequenced, yielding 220,000-530,000 5′-clipped cDNAs per library.

7.1.2 Mapping and basic sequence analysis

After 454 sequencing the reads of all libraries were mapped using segemehl. Disregarding cDNAs <12 bp, 62.7%-84.2% of the cDNAs in each library could be mapped to the H. pylori chromosome, and 25.7%-75.2% of the reads mapped uniquely to the genome. The longest mapped cDNA reads were ~350 nt. For all hits and for all reads the scores were summed up across the reference genome. The fold coverage of the whole genome is then expressed as mean score for all nucleotides. The fold coverage for protein coding sequences, RNAs or other genomic subsets was calculated equivalently. For example, the fold coverage of coding sequences is given by the mean score for all nucleotides within such coding sequences. While about 40% of the whole genome is covered at least 5-fold, >70% of the mRNAs are covered at the same level (Tab. 7.1). Overall genome fold coverage for all libraries was calculated separately for the plus strand, the minus strand, and the sum of both strands.

If expression occurred on both strands at some genomic position, the weaker expression was declared as antisense (Fig. 7.1). Approximately

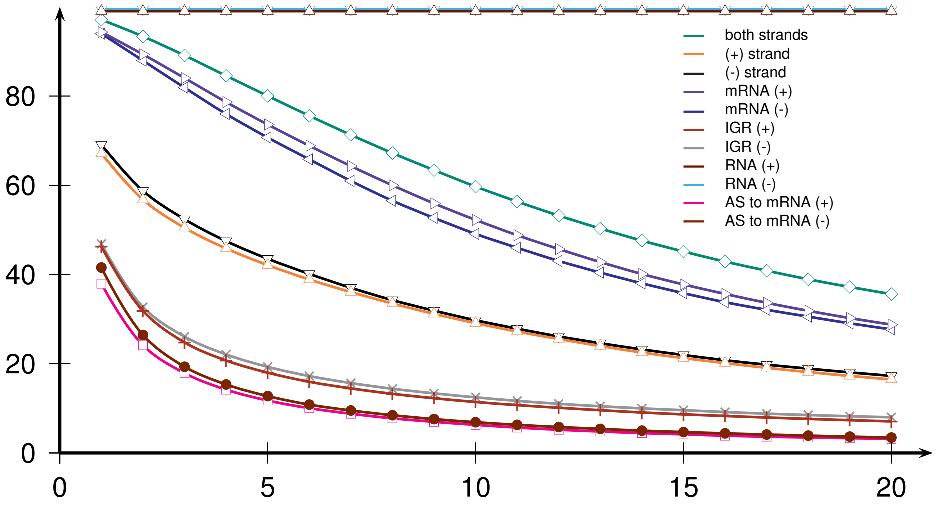


Figure 7.1: Coverage of the whole H. pylori genome. About 70% of the nucleotides on either strand of the H. pylori genome are covered by normalized cDNA sequence hits (curves with triangles up and triangles down). The minimum fold coverage of nucleotides in protein coding sequences (triangle right and triangle left) and annotated RNA (straight) is substantially higher: >95% of all known functional sequences have a minimum fold coverage of 1 and >50% are covered at the 10-fold level. The minimum antisense fold coverage of 5 for about 10% of mRNA sequences indicates substantial antisense transcription (squares and circles). Additionally, the minimum fold coverage of intergenic regions (IGR; stars and crosses) as well as the sum of minimum fold coverages of both strands (diamonds) are shown.

70% of the nucleotides on both the leading and lagging strand were covered ⩾1-fold. As expected for the whole transcriptome of H. pylori, fold coverages on both strands are inversely proportional. Hence, single coverage for base pairs rises to 97%, and even 80% of the genome was covered at least 5-fold. The coverage of annotated mRNAs is equivalently high: >70% of the nucleotides within mRNA annotations are ⩾5-fold covered (Fig. 7.1). Interestingly, ~10% of annotated mRNAs show antisense transcription at the 10-fold level. Figure 7.2 shows a detailed distribution of reads ⩾12 bp for each library, reads with no match to the H. pylori genome, or reads that overlap with annotated regions. The fraction of reads that could not be mapped to the Helicobacter genome is higher for the AG-/+ and HU^{-}/$^{+}$ libraries as they also contain human reads from the host cells. The fraction of uniquely mapped reads is generally higher in the enriched libraries, due to the removal of a large number of ribosomal rRNA reads. Reads which derive from these transcripts map at least twice to the genome due to the presence of two 16S and 23S rRNA genes and three 5S rRNA genes. The fraction of ribosomal RNAs is reduced in the enriched (+) libraries versus (-) libraries (Fig. 7.2). For example, in the ML- library ~63% of all reads derive from rRNAs, but only 25% in the ML+ library. Conversely, the fraction of tRNAs increased upon TEX treatment (7.3% tRNA reads in the ML- versus 27% tRNA reads in ML+), for

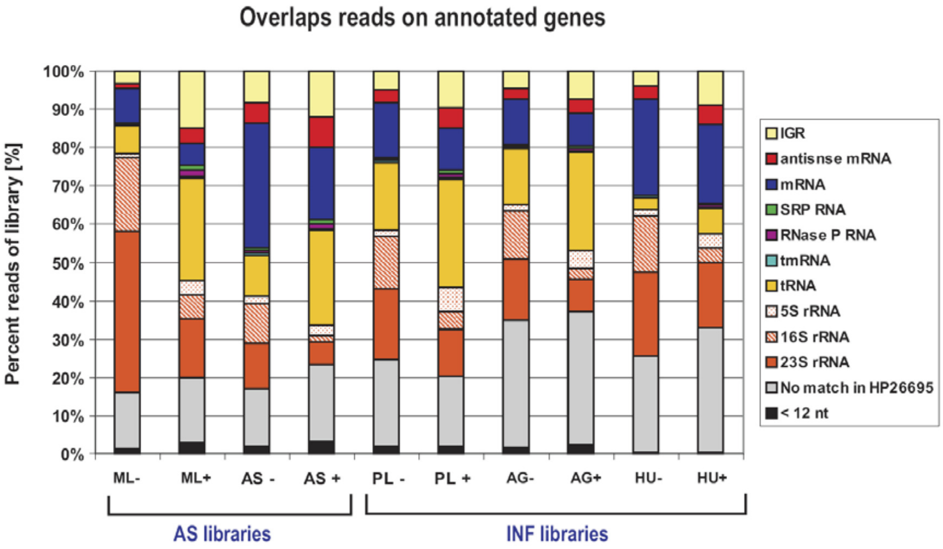


Figure 7.2: Read distribution over annotated genes. Bar diagrams showing relative proportions of various RNA classes in the dRNA-seq libraries

reasons currently unknown. A large fraction of reads mapped to mRNAs (8.8% / 5.8% of the ML- / ML+ libraries), as is most evident in the acid stress libraries (32.5% / 18.9% in AS- / AS+). Moreover, a greater fraction of reads mapping antisense to annotated genes was observed (5.4% / 7.9% in AS- / AS+). These reads probably derive from cis-encoded antisense RNAs. In addition, the fraction of reads from intergenic regions contains promising candidates for novel H. pylori sRNAs (e.g., 3.4% / 15% in ML- / ML+). Furthermore, the increased read number from intergenic regions in the (+) libraries shows the successful enrichment of primary transcripts upon TEX treatment.

7.1.3 Identifcation of start sites

The transcription of a gene may start at different sites in the genome. We thus define a “Primary TSS” as the TSS that has the highest cDNA coverage within a distance ⩽500 bp upstream of annotated ORFs or mature 5′ ends of small RNAs; secondary TSS being associated with the same gene as a nearby primary TSS but showing fewer cDNA reads; internal TSS located within an annotation on the sense strand; antisense TSS situated inside or within ⩽100 bp of an annotation on the opposite strand; and orphan TSS if there is no annotation in close proximity. For the TSS detection the reads from the libraries for all conditions were merged into a TEX+ and TEX- set. The overwhelming majority of TSS could be determined by the following automated procedure. Transcription signals were compared between enriched and non-enriched libraries. A TSS was proposed automatically if the signal in the enriched library was 2-fold higher compared to its non-enriched counterpart and a sudden increase of coverage by $\geqslant 5$-fold was detected within a window of 20nt downstream of the potential TSS (compared to the window of 20nt upstream of the potential TSS). Approximately 95% of the predictions of this simple linear time algorithm were validated by two independent assessors. In the context of this work a statistical model for the refinement of this simple approach providing a probability measure was developed based on the Skellam distribution.

Definition 22 (Skellam) *The probability mass function for the difference* $D = X - Y$ *of two random poisson distributed variables* X *and* Y *with the arrival rates* λ_1 *and* λ_2 *respectively is given by*

$$F(k, \lambda_1, \lambda_2) = e^{-(\lambda_1 + \lambda_2)} \left(\frac{\lambda_1}{\lambda_2}\right)^{k/2} I_{|k|}(2\sqrt{\lambda_1 \lambda_2}) \tag{7.1}$$

where $I_{|k|}$ *denotes a first rank Bessel function of* $|k|^{th}$ *order.*

The basic idea behind this approach is that the rate of "read arrivals" in the non-TEX libraries is increased compared to the TEX-treated libraries. One of the major challenges is to determine the λ-values for TEX+ and TEX- libraries. While the model can be applied to transcribed regions it is not applicable to untranscribed regions and thus requires a partition of the genome prior to the evaluation. In Schmidtke et al. a solution to the problem was found by evaluating local and global arrival rates. Here, for each libary a global λ_{global} and for each local window of a constant size λ_{local} is evaluated. The quotient

$$\lambda = \frac{\lambda_{\text{global}}}{\lambda_{\text{local}}}$$

provides a feasible value for the automatic detection of transcription start with probability measures. The automated approach does not distinguish between primary TSS and processing sites with sufficient accuracy. Thus, the issue of automated TSS detection requires further attention.

In a few cases of the Helicobacter pylori transcriptome (<2% of all annotated TSS) a clear enrichment was only seen for one condition leading a low coverage insufficent for the simple automatic detection strategy. We thus included other criteria. For example, two divergently transcribed genes, i.e. one gene encoded on the lagging strand and the neighboring gene in the genome is encoded on the leading strand, can have overlapping promoter regions if they are encoded in close vicinity. In this kind of genomic context, both genes need their own promoters for transcription. We annotated these 5′ ends as TSS based on their location. In total, there are almost equal TSS numbers from the lagging (n=969; ~51%) and

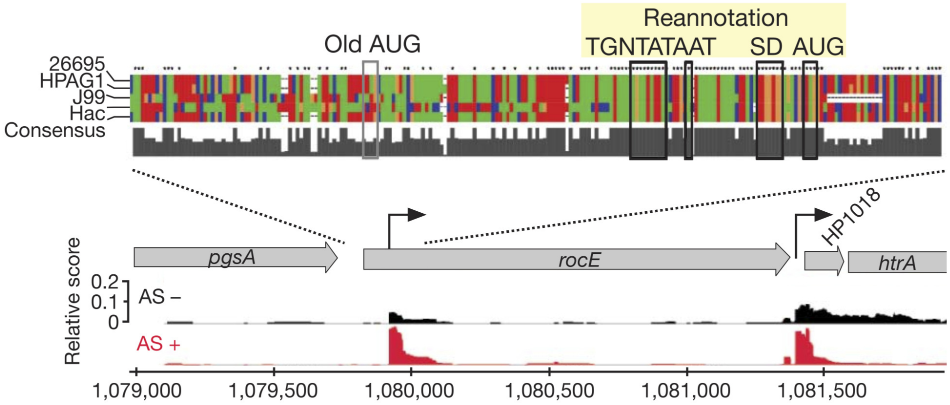


Figure 7.3: Correction of the rocE annotation of Helicobacter pylori by dRNA-seq. Since the transcription start site of rocE is found downstream of the orignially annotated start codon (AUG) it was possible to reannotate the translation start and its Shine-Dalgarno sequence, an important ribosomal binding motif, to a highly conserved region about 30bp downstream of the old AUG.

leading (n=938; ~49%) strand of H. pylori. A complete list of all TSS of Helicobacter pylori is given in [Sharma 2010]. Several gene annotations were corrected based on the dRNA-seq analysis. In the case of the rocE gene, an arginine permease involved in amino acid transportation, the dRNAseq-inferred transcription start site helped to reannotate the translation start site approximately 30 bp downstream of the original AUG-start codon (Fig. 7.3). Besides rocE, 18 more genes were corrected.

The highly expressed non-coding 6S sRNA, a riboregulator of RNA polymerase (RNAP), was found in enriched and non-enchriched libaries. This sRNA escaped all previous analyses possibly due to the annotation of a poorly conserved hypothetical protein on the opposite strand. Due to the high sensitivity of the segemehl aligner that is capable of aligning extremely short reads of 12nt it was possible to also find the pRNAs (Fig. 7.4), that are produced by the RNAP using the 6S RNA as a template [Cavanagh 2010, Trotochaud 2005, Wassarman 2000].

In other cases it was possible to reveal complex operon structures. In the case of the cag pathogenicity islands, a new sub-operon, especially expressed under acid stress conditions, was found. The TSS for this new operon is found within the the cag23 protein coding sequence. This new finding explains a previously reported [Wen 2003] selective up-regulation of cag22-cag18 that occurs independent from the primary cag25-18 operon under acid stress. This can also be validated by the fold-enrichment analysis performed in the context of this work. While cag22 shows a 10-fold enrichment under acid stress cag23 is merely enriched 5-fold (cf. Tab. 7.2). Three start sites in the cag25-cag18 operon were independently verified by 5′-RACE experiments (Fig. 7.5) In total, 69 of the TSS identified in this work have been analysed using alternative methods. The comparision de-

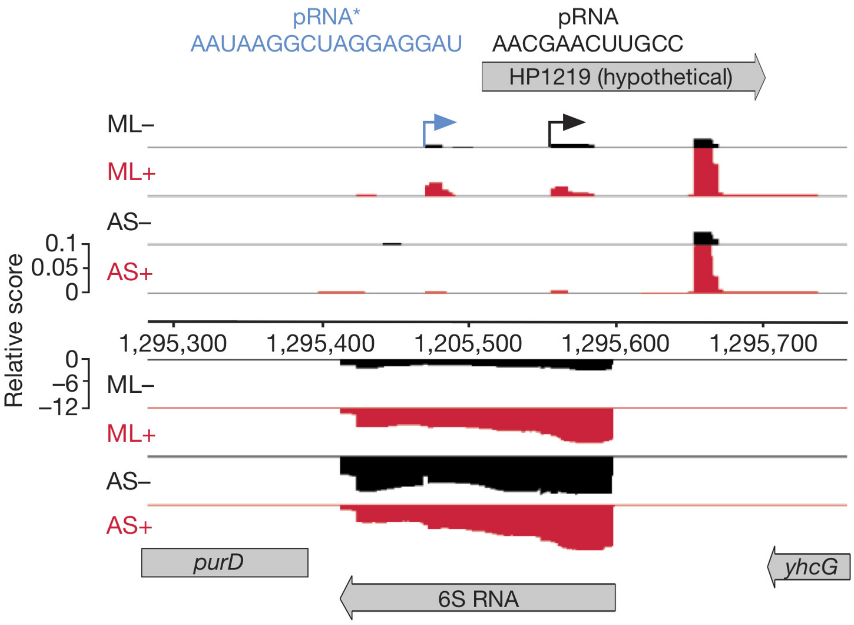


Figure 7.4: The H. pylori 6S RNA can be found in ML and AS libraries. With a length of 180nt it is found opposite to the hypothetical protein HP1219. The 6S RNA-specific pRNA/pRNA* including their 6S RNA-borne TSS are shown at the top.

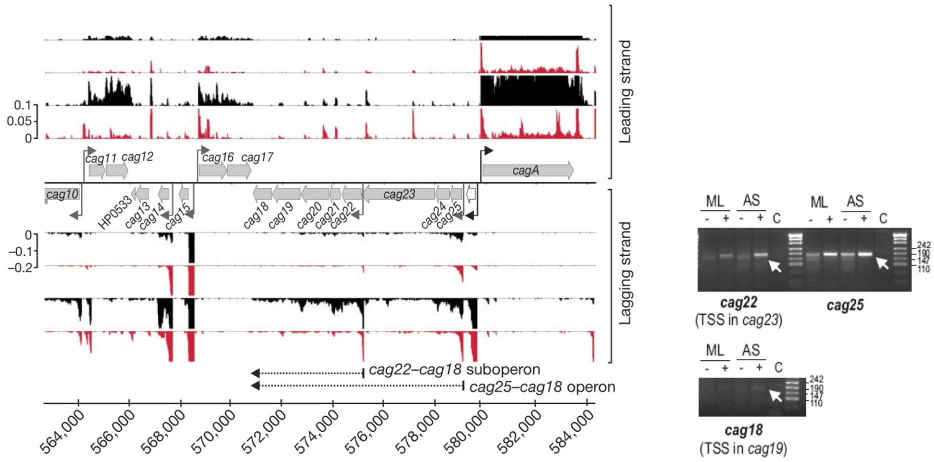


Figure 7.5: The dRNAseq analysis reveals a more complex transcriptional organization. The presence of the TSS upstream of the cag25-cag18 primary operon can be confirmed by 5′ RACE. Additionally dRNAseq captured the predicted acid-induced TSS in cag19 which drives monocistronic transcription of cag18. The dRNAseq signal as well as the 5′RACE product however is weak in this last case.

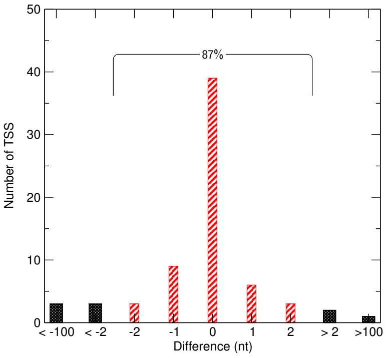


Figure 7.6: Histogram indicating observed distances between 69 TSS identified by dRNA-seq vs independent experimental approaches. Of these TSS 87% matched within 2 nt tolerance. This analysis includes experimentally tested TSS as well as TSS taken from the literature. It is important to notice that all 22 TSS 5′RACE experiments carried tested in the context of [Sharma 2010] match within 1 nt tolerance.

picted in Fig. 7.6 shows the high accuracy of the dRNAseq protocol and evaluation methods used here. About 90% of the experimentally validated results did not differ by more than 2nt.

The 5′UTR (untranslated region) ranging from TSS to start codon determines the translational efficiency of messengers. In striking accordance with the structurally observed ribosome contacts [Ramakrishnan 2002], our TSS annotation reveals that most (~50%) of the 5′UTRs are 20-40 nucleotides in length and support the AAGGag motif [Vanet 2000, Yada 2001] located about 6nt upstream of the start codons. After correction for multiple testing using the FDR method [Benjamini 1995], we found correlations of 5′UTR length with cellular function. For example, nucleotide- and nucleoside-related genes almost invariably have ~30 nt-long 5′UTRs, as if optimized for translation. In contrast, regulatory genes of cellular processes such as cell division, pathogenesis or transformation possess significantly longer 5′UTRs than the rest of the genes (Fig. 7.7).

Although leaderless mRNAs are considered rare and primarily phage-associated in Gram-negative species [Brock 2008], we found that ~2.2% of all H. pylori mRNAs have a 5′UTR <10 nt. At 26 genes, including the dnaA, recR and hemH housekeeping genes, transcription initiates exactly at an AUG start codon essential for stable ribosome binding of leaderless mRNAs.

7.1.4 Differential expression analysis

To compare expression at the TSS in the TEX-treated cDNA libraries of the five different growth conditions, the dRNA-seq mapping data of each enriched library (ML+, AS+, PL+ AG+, and HU+) was normalized to the number of mapped nucleotides for each library. A window of

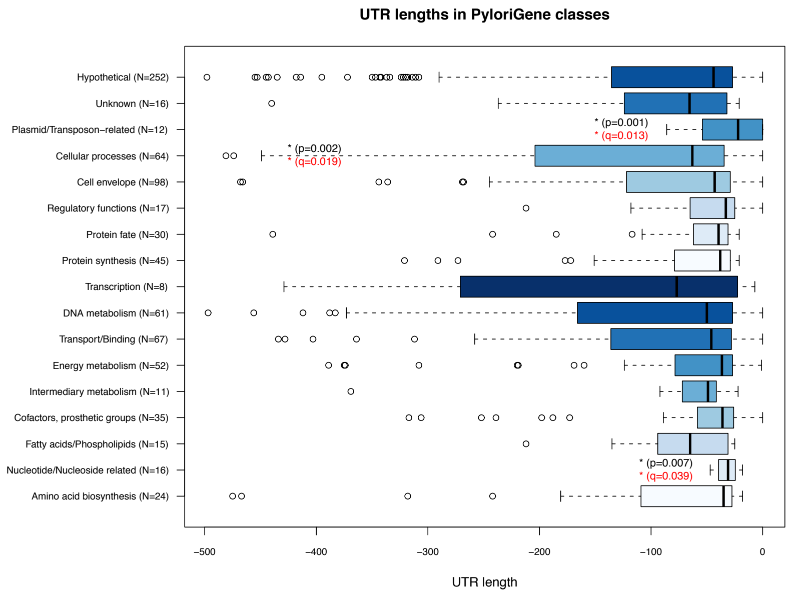


Figure 7.7: UTR length box plots for the categories of the PyloriGene functional classification system. Each UTR was associated with the functional classes defined at PyloriGene database of its downstream gene and subsequently grouped by function. The box plots show the distribution of UTR lengths in 17 different functional classes in which the box indicates the range from the lower to the upper quartile. A line in the middle of each box indicates the median. Whiskers indicate the largest and smallest non-outliers of the data set. Extreme outliers are depicted by circles. A non-parametric Wilcoxon test was applied to check for significant UTR length deviations among classes. In total, three classes show differences ($p < 0.05$) that remain significant after fdr adjustment for multiple testing ($q < 0.05$). While plasmid- and transposon-related genes have particularly short UTRs, genes involved in the regulation of cellular processes such as cell division, pathogenesis or transformation show significantly increased UTR lengths. Interestingly, UTRs of nucleoside- and nucleotide-related genes populate a very narrow band around 30nt without any outliers.

40 nt was anchored downstream of all primary TSS. The fold-coverages within the TSS-window were used to compare the promoter expressions in the enriched acid-stress and infection libraries in a semi- quantitative way. For each pairwise comparison a minimum normalized fold-coverage (normalized to the total genome expression of each library to be able to compare libraries with different mapped read numbers) of 1.5e-10 (~10-fold coverage in the ML libraries) was required in at least one library. Note that only genes with primary TSS and satisfying the coverage threshold were considered. Consistent with the expression differences observed for the whole urease mRNAs given in Table 7.2, a >2-fold up-regulation of expression at the ureI and ureA TSS was observed upon acid stress. Furthermore, additional detectable changes in TSS coverage upon acid stress as well as in the presence of eukaryotic

cells could be observed. For example, we confirmed the previously reported down-regulation of HP1177 (omp27) and HP1212 (ATP-synthase subunit C) upon acid stress [Bury-Moné 2004] , and of HP0682, HP0974, and HP0218 upon contact with AGS cells [Kim 2004] . Although this data is only semi-quantitative, it indicates that dRNA-seq can potentially be used for gene expression profiling, more replicates and higher cDNA coverage for statistical analysis permitted.

7.2 An insight to post-transcriptional modification

Many aspects of the RNA maturation leave traces in RNA sequencing data in the form of deviations from the reference genomic DNA. This includes, in particular, genomically non- encoded nucleotides and chemical modifications. The latter leave their signatures in the form of mismatches and conspicuous patterns of sequencing reads. Modified mapping procedures focusing on particular types of deviations can help to unravel post-transcriptional modification, maturation and degradation processes. In the context of [Findeiss 2011], we focus on small RNA sequencing data that is produced in large quantities aimed at the analysis of microRNA expression. Starting from the recovery of many well known modified sites in tRNAs, the work provided evidence that modified nucleotides are a pervasive phenomenon in these data sets. Regarding non-encoded nucleotides we concentrate on CCA tails, which surprisingly can be found in a diverse collection of transcripts including sub-populations of mature microRNAs. Although small RNA sequencing libraries alone are insufficient to obtain a complete picture, they can inform on many aspects of the complex processes of RNA maturation. Disregarding technical artifacts in the RNA sequence read and errors or missing data in the reference genome, which make it impossible to map the read at all, there are at least three reasons why RNA reads do not match exactly to the reference genome: (i) sequencing errors, (ii) polymorphisms, and (iii) RNA maturation. Hence, analysis of RNAseq data in general requires a significantly higher sensitivity in comparison to DNA variation analysis. Thus, a sensitive mapping algorithm is key to identify these in HTS data accordingly.

7.2.1 Short RNA sequence data

We use a combination of small RNA libraries, sequenced on an Illumina platform, from human and Rhesus macaque brains, respectively [Somel 2010]. The human library comprises 71 307 445 sequencing reads, with an average length of 22 nucleotides. Having 114 619 534 reads the macaque data set is approximately twice as big as the human one and has a similar read length distribution, with an average

HP number	Gene	Start	End	Strand	AS-/ML-	AS+/ML+
HP0067	ureH	71968	72818	-	1.7	0.8
HP0068	ureG	72818	73417	-	3.4	2.7
HP0069	ureF	73446	74210	-	7.3	1.1
HP0070	ureE	74233	74745	-	4.3	1.2
HP0071	ureI	74747	75398	-	3.3	3.7
HP0072	ureB	75445	77236	-	2.8	1.3
HP0073	ureA	77240	78012	-	4.3	2.6
HP0520	cag1	547272	547675	+	5.9	2.1
HP0522	cag3	548134	549579	+	2.8	8.0
HP0523	cag4	549589	550098	+	5.4	2.3
HP0524	cag5	550217	552463	-	3.6	10.8
HP0525	virB11-like	552472	553464	-	5.9	8.3
HP0526	cag6	553469	554179	-	4.8	3.1
HP0527	cag7	554206	559989	-	3.0	6.6
HP0528	cag8	560004	561595	-	4.8	5.9
HP0529	cag9	561625	563232	-	6.3	3.2
HP0530	cag10	563237	564078	-	3.1	7.1
HP0531	cag11	564347	565037	+	5.0	2.1
HP0532	cag12	565073	565915	+	6.3	4.6
HP0533	hypothetical	566035	566124	-	2.0	70.1
HP0534	cag13	566126	566780	-	2.5	4.1
HP0535	cag14	567142	567666	-	5.8	3.9
HP0536	cag15	567955	568299	-	7.2	5.0
HP0537	cag16	568686	569853	+	4.2	4.4
HP0538	cag17	569868	570788	+	3.0	0.5
HP0539	cag18	570870	571583	-	34.6	4.0
HP0540	cag19	571580	572725	-	14.6	6.0
HP0541	cag20	572736	573848	-	5.0	1.7
HP0542	cag21	573864	574292	-	9.5	3.7
HP0543	cag22	574347	575200	-	10.8	11.5
HP0544	cag23	575155	578106	-	4.7	7.0
HP0545	cag24	578115	578738	-	4.3	6.6
HP0546	cag25	578740	579114	-	2.1	1.5
HP0547	cag26(cagA)	579817	583481	+	4.2	2.0

Table 7.2: Fold enrichment in cDNA coverage in the urease operon and cag island. Fold enrichment in cDNA coverages upon acid stress induction for genes of the urease operon and cag island were calculated. Specifically, for each gene (including 5′UTR or 3′UTR for first/last genes within operons) the total number of mapped nucleotides were calculated and normalized to the total number of mapped reads for each library. Afterwards, the enrichment in cDNA coverage under acid stress condition was calculated as the ratios for the TEX- libraries (AS-/ML-) and the TEX+ libraries (AS+/ML+).

		H. *sapiens*	M. *mulatta*
Entire library			
	Reads	71 307 445	114 619 534
	Tags	355 453	14 240 332
3-′CCA			
	Tags	3 925	138 895
	NUMT (-)	3 017	118 298
Non-genomic 3′-CCA			
	Tags	1 431	90 208
	Blocks	246	1 289

Table 7.3: Summary of the analysed short read libraries for detection of post-transcriptional regulation. Reads with identical sequences were merged into tags. All tags matching the mitochondrial genome of human and macaque, respectively, were removed to avoid contamination of nuclear copies of mitochondrial DNA (NUMT). Overlapping tags that survived the filtering steps were joined into blocks.

of 23 nucleotides. To allow error-tolerant mapping of cDNA sequences ($\geqslant$14 nucleotides in length), segemehl was used with standard parameters and an E-value cut-off of 500 to also align short sequences. To measure the activity of nucleotidyltransferases, tags ending with 3′-CCA were selected. The CCA was removed and the truncated tag was mapped to the reference genome. Tags with a genomically encoded CCA end downstream of the mapped tag were excluded from further analysis. Because short reads deriving from nuclear copies of mitochondrial DNA (NUMT) [Hazkani-Covo 2010] and reads truly deriving from the mitochondrial DNA can not be reliably distinguished in the data at hand, all tags matching to the mitochondrial genome of the respective species have been excluded. Overlapping tags passing the filtering steps were then joined into blocks [Langenberger 2009]. Finally, blocks representing less than 10 reads were excluded from further analysis. In our analysis, we distinguish between individual reads and tags. A tag is defined as a DNA sequence that occurs at least once in a set of sequencer reads. Thus, a tag typically corresponds to several identical reads. The advantage of using tags is that there is a drastic reduction in the amount of data to be handled. A statistical overview of the analyzed data sets is given in (Tab. 7.3).

7.2.2 Chemical modifications

Some chemical modifications of nucleotides are detectable as mismatches between RNAseq data and the genomic reference. In contrast to PCR artifacts, the mismatches appear in many different tags and the frequency distribution of nucleotides deviates from that expected for SNPs. Two recent studies showed that tRNA modifications are detectable in plants [Ebhardt 2009, Iida 2009]. This is also the case in mammals, for example, the well-known 1-methyladenosine modification is found at position 58 of many tRNAs (Fig. 7.8) [Roovers 2004]. The 1-methyladenosine modification is pivotal for the stability and thus the function of tRNAs [Anderson 1998]. It has been reported that the methylated adenosine residue 58 serves as a pause signal for plus-strand strong-stop DNA synthesis and termination site during reverse transcription [Renda 2001]. A closer look at the data (inset in Fig. 7.8) shows that this particular modification of adenine is typically interpreted by the sequencer as an A-to-T transversion or an A-to-G transition. Strikingly, the substitution matrix for this modification appears to be largely invariant to the library preparation. This is the most prominent modification that is directly visible from the superposition of the error profiles of all tRNAs. Several other modifications are detectable as conspicuous accumulations of mismatches in individual tRNAs. Notably, most of the detectable variations are located either towards the 5′- or the 3′-end of the tRNA. This is caused by the very uneven coverage of tRNAs with by short sequencing reads, which is heavily biased towards the ends and the fact that the error-prone 3′- and 5′-termini of sequencing reads naturally coincide with the ends of the tRNA. The genomic distribution of mismatches that can be interpreted as sites of chemical modifications are summarized (Fig. 7.9). As a result of the short RNA sequencing protocol tRNAs and miRNAs, as measured by the proportion of their genome sequence, accumulate most of the sequence variations observed in the data sets used here. For tRNAs, the numbers

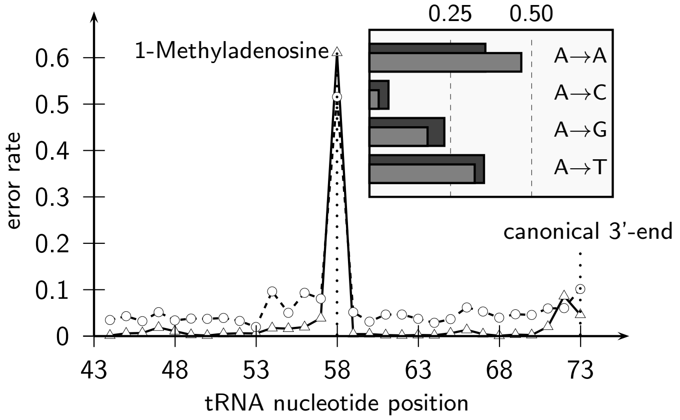


Figure 7.8: Frequency of mismatches between RNAseq reads and genomic sequence for reads mapping close to the 3′-ends of human (solid line, triangles) and macaque (dashed line, cirlces) tRNAs. Reads are aligned at the tRNAse Z cleavage site, so that tRNA position 58 corresponds to position -16 in the alignment. The inset shows the frequency of nucleotides observed at this position. Apart from the being read "correctly" as A, this post-transcriptional modification is typically seen as an A-to-T transversion by the sequencer. Human (dark gray bars) and macaque (light gray bars) show similar substitution patterns.

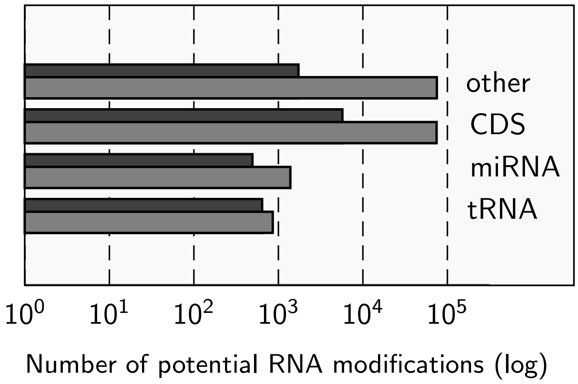


Figure 7.9: Distribution of potential modifications among human (dark gray) and macaque (light gray) RNA classes. Despite the fact that the data set generated in the macaque RNAseq experiments is approximately 50-times larger than the human one (Table 1), the total number of possible modifications in tRNA and miRNA does not differ proportionally. In the case of tRNAs, 862 modifications were found in macaque compared to 657 in human sets; in the case of microRNAs, there are 1400 potential modifications in macaque and approximately 500 in human.

are comparable between human and macaque (657 versus 862, respectively) indicating that even the comparably small human library is nearly saturated. Whereas for microRNAs, there are approximately 500 positions with significant sequence variation in human and 1400 in macaque, leading us to expect that the true number of putative modifications in human microRNAs will be larger than that observed here. As a consequence of the small RNA sequencing protocol, such a saturation effect can not be expected for long transcripts, including coding sequences (CDSs). The much higher difference of observable nucleotide variations within these classes clearly supports this assumption. Surprisingly, more than two-third of the detectable loci in human map to coding sequences, whereas only 8% map to tRNAs. In macaque the ratio is even smaller, suggesting that chemical modifications indicated by the observed sequence variations are generally a common phenomenon. A detailed analysis of individual modification sites with respect to their substitution patterns requires a sufficiently large coverage.

7.2.3 Non-templated CCA-additions

tRNA biogenesis involves multiple maturation steps of the primary RNA polymerase-III transcripts including removal of the 59-leader, trimming of the 3′-trailer, addition of CCA, splicing of introns that could be present and chemical modification of multiple nucleoside residues [Hartmann 2009, Phizicky 2010]. Enzymatic CCA addition by nucleotidyltransferases is considered essential for tRNA biosynthesis in all organisms that do not encode CCA termini at the genomic level. These en-

	Human		Macaque	
	3′-CCA blocks			
tRNA	99	40.2%	219	17.0%
miRNA	52	21.1%	111	8.6%
U2	3	1.2%	3	0.2%
rRNA	2	0.8%	0	0.0%
CDS	2	0.8%	2	0.1%
pseudo ncRNA	10	4.1%	13	0.9%
unknown	79	32.1%	941	73.9%
TOTAL	246		1289	

Table 7.4: Distribution of blocks with non-genomically encoded CCA tails in different RNA classes. RNA classification was derived from the tRNAscan-SE prediction and the downloaded RNA gene track. Pseudo ncRNAs lack important features (e.g. parts of the characteristic secondary structure) of their functional siblings. A significant proportion of CCA tags was expectedly found in tRNA loci for both human and macaque.

zymes can add specific nucleotides or nucleotide sequences in the absence of genetic templates [Xiong 2004]. Although CCA nucleotidyltranferases (also known as CCA-adding enzyme) primarily recognize and process tRNAs [Vörtler 2010], they are also known to have non-tRNA substrates. Many of them are tRNA-like elements such as the mascRNA processed from the 3′-terminus of MALAT1, a long non-coding RNA that is known to be misregulated in many human cancers [Wilusz 2008] and viral RNA motifs that act to attract the hosts processing machinery [Bogerd 2010]. Another prominent example is the U2 small nuclear RNA (snRNA), which undergoes a maturation process similar to that of tRNAs. After removal of a 3′-trailer by a 3′ exonuclease, the trimmed U2 snRNA is processed by the human CCA-adding enzyme [Cho 2002]. CCA addition has also been observed for several mitochondrial mRNAs [Williams 2000]. These examples of non-standard 3′-CCA tails are also visible in the RNAseq data sets for both human and macaque (Tab. 7.4). We therefore asked whether additional substrates of the CCA nucleotidyltranferase can be identified. A large fraction of the tags match mitochondrial sequences. After removing those, we retained 246 blocks in human (and 1289 blocks in the much larger macaque library). The moderate 5-fold increase of CCA blocks in rhesus monkey compared to human, despite a 50-fold higher coverage in the library, suggests that the detected set of CCA-tagged RNAs approached saturation in the monkey library. It can only be speculated that, for the same reason, a majority of the 73.9% CCA blocks map to unknown locations in macaque, whereas only 32.1% of human CCA blocks fall into the same category. As expected, a substantial fraction of blocks (and the majority of reads and tags) belong to tRNAs (Tab. 7.4). However, there are a large number of microRNAs with non-genomically encoded CCA ends. A clearly successful in-vitro verification with an autoradiographic detec-

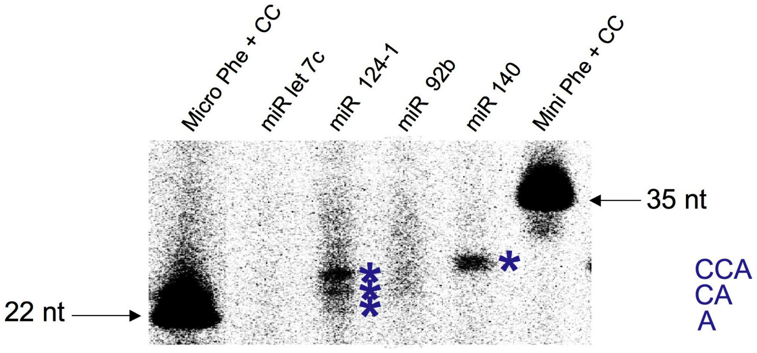


Figure 7.10: In-vitro verification for single stranded miR CCA-addition. Autoradiographic detection of PAA gels (containing 7 M Urea) from nucleotide incorporation Assay with 2 μCi [α-32P]-ATP. Assay with 5 pmol of microRNAs, 200 ng hmCCase in the presence of GTP, UTP, CTP (1mM each), ATP (100 μM), respectively.

tion method was possible for the single stranded human miRNA 124-1 and the miRNA-140 incubated with a nucleotidyl-transferase and hot ATP (Fig. 7.10). In contrast to the observation in the RNA sequencing experiments, a verification of the miRNA let-7c failed. While the bands in the experiment clearly show three lanes, potentially caused by sequential A, CA, and CCA additions, the miRNA 140 shows a single strong band. Surprisingly, all double-stranded miRNAs (RNA/DNA hybrid) tested show nucleotide incorporation (Fig. 7.11). It is unclear though if the differences of the band positions for double stranded miRNAs are caused by sequence-specific gel mobilities or different nucleotide additions.

7.3 Mutated splice sites in Lymphoma samples

The Burkitt lymphoma is a aggressive cancer derived from germinal B-cells and the most frequently occurig B-cell lymphoma in children [Swerdlow 2008]. In recent years several chromsomal translocations namely t(8;14)(q24;q32), t(2;8) and t(8;22) have been identified to affect the regulation of the c-myc gene, critical to the development of Burkitt lymphoma. The translocations juxtapose myc to immunoglobulin loci [Boerma 2009] and thus cause a massive dysregulation of its expression. In a study conducted within the ICGC-MMMLSeq consortium we have identified a novel gene potentially involved in the pathogenesis of this disease using whole exome-, whole genome and transcriptome sequencing [Richter 2012]. The method for splice site detection described above significantly corroborated the results of this study. In a first step SNVs were detected in whole-exome and whole-genome data of four different patients suffering from Burkitt lymphomas. In total approximately 2500 somatic aberations were identified using a pipeline established at the Deutsches Krebsforschungszentrum (DKFZ) in Heidelberg. Per case

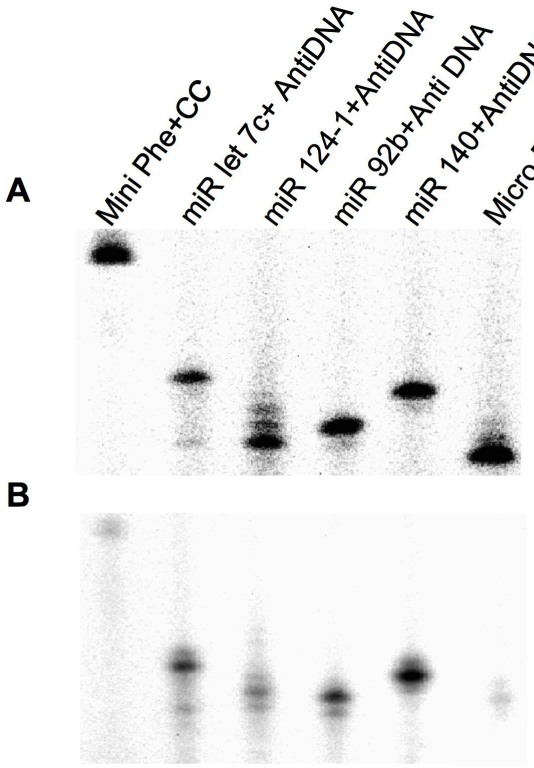


Figure 7.11: In-vitro verification for double stranded miR. Autoradiographic detection from nucleotide incorporation Assays with 2 μCi [α-32P]-ATP and 5 pmol RNA/DNA hybrids, 200 ng hmCCase in the presence of: (A) all NTPs (10 mM each), and (B) GTP, UTP, CTP (10 mM each) + ATP (100 μM).

around 30 of these mutations have been identified to be protein-changing, in total this amounted to a total of 134 non-synonymous mutations. Analysis of the transcriptome using the read aligner described above, we identified a 63 of these genes to be expressed where the mutated allele was expressed in 57 cases. Focussing only on those genes that were mutated in more than one patient we identified 7 frequently mutated and expressed genes. Namely, TP53, MYC, SMARCA4, FBXO11, DDX3X, RHOA and ID3 [Wilda 2004, Johnston 1992, Wang 2011, Duan 2012]. While the first three genes have already been described to be involved in the development of Burkitt Lymphoma, the other four genes have only been linked to other types of B-cell lymphoma. The gene ID3 was found to be mutated in two patients. The patients shared a non-sense as well as a splice site mutation. In fact, the splice site mutations resulted in aberant splicing of the ID3 genes as confirmed by the described splice site detection algorithm (Fig. 7.12). In a verification using BL4 cell lines the aberant splicing was confirmed.

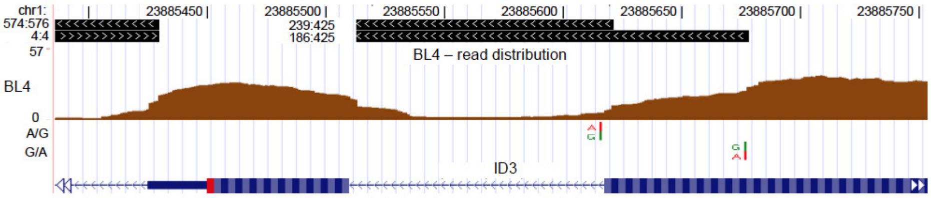


Figure 7.12: Biallelic splice site mutation of the ID3 gene in patient BL4. The mutation causes aberant splicing. Two abberant splice products, non of them at the annotated wild type splice site, can be observed in the transcriptome of patient BL4.

7.4 Short read alignments to repetitive regions

A recent study suggests that repetitive elements make up over two-thirds of the human genome [de Koning 2011], while previous studies have reported a significantly lower repeat content [Batzer 2002]. The authors of the same study suggest that transposable elements have played a large role in the shaping of the human genome. Several of these elements, namely the family of Alu elements are comparably young, specific to primate genomes and thus especially well suited to study human evolution and diversity [Rowold 2000]. Several Alu repeats are sources of non-coding RNAs that are exclusively found in human genomes, such as the brain specific BC200 RNA [Amaral 2008]. It has been noted that the many transposon-derived elements are in fact functional (cf. [Amaral 2008]). The novel sequencing technologies for the first time offer to study whole genomes and transcriptomes and thus provide insight to the human repeatome. To do this, however, it is necessary to design mapping and assembly algorithms that are capable to identify and resolve the conflicts that may arise during assembly and alignment of sequencer reads. The mapping algorithm devised in this thesis has been applied to *chromatin immuno precipiation sequencing* (ChIP-Seq) data. Specifically, in this study the chromatin modifiying effects of a human long non-coding RNA (lncRNA) located on Chr9p12 antisense to the INK4 locus (ANRIL) were investigated. Chr9p12 is presumably involved in the genetic predisposition to atherosclerosis and is the strongest known genetic risk factor for heart attacks and strokes [Helgadottir 2007, McPherson 2007].

Previous studies showed that ANRIL may excert its function by binding to and regulating chromatin modifiers, specifically SUZ12 and CBX7, that are part of the Polycomb repressive complex (PRC) [Yap 2010, Kotake 2011]. This complex is known to work by methylation, demethylation and acetylation of histones. In [Holdt 2013] a set of highly differentially regulated genes upon ANRIL overexpression was reported. Subsequently, we were able to find an ALU-DEIN motif present in the ANRIL gene itself as well as the promotor regions of the differatially regulated genes. In total the motif was present in approximately 60000 copies in

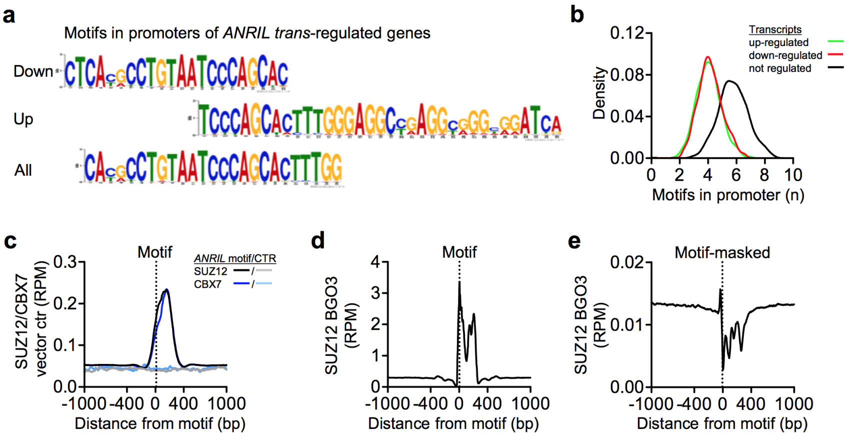
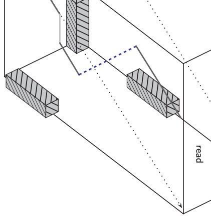


Figure 7.13: Motif search and ChIP-Seq results with and without multiple hits. (a) motifs that were identified using the MEME package in up-, down- and all differentially regulated genes all share the same common core motif that is mirrored in the ANRIL lncRNA sequence. (b) In promotors of differentially regulated genes the ALU-DEIN motif occurs less frequent than in promotors of ANRIL-independent genes. (c) In negative conrol DNA experiments a small peak of 0.2 reads per million aligned reads (RPM) is observed downstream of the motif in differentially regulated promotors, while in ANRIL overexpressing cell lines a peak of more than 3 RPM can be observerd directly at the motif (d). This motif peak vanishes if multiple matches are not allowed (e).

the whole genome (Figure 7.13a). Interestingly, the motif occurs less frequently in unregulated genes compared to ANRIL-regulated genes (Figure 7.13b). The immediate question arising from this finding was whether histone modifications occur in the close vicinity to the newly identified ALU-DEIN motif. To do that a genome wide mapping of SUZ12 and CBX7 occupancy was performed in ANIRL-overexpressing cell lines and normal controls using ChIP-Seq. Using the read mapping algorithm described above, notably its capability to report all multiple hits up to the user given maximum occurence parameter. While there is a mild increase of mapped reads downstream of the identified motif in promotors of differentially regulated genes (Fig. 7.13c) in cell lines that do not over-expres ANRIL, a strong peak of more than 3 reads per million aligned reads (RPM) can be observed directly at the motif positions of differentially regulated promotors for SUZ12 (Fig. 7.13d). This observation, however, can only be made if the read aligner is able to report all multiple hits. If not, the signal peak around the motif may not be detected (Fig. 7.13e).



Chapter 8

Discussion

The successful mapping of high-throughput sequencing (HTS) reads to reference genomes largely depends on the accuracy of both the sequencing technologies and reference genomes. We have developed and implemented a new approach for short read mapping that, in a first step, computes exact matches of the read and the reference genome. The exact matches are then modified by a limited number of mismatches, insertions and deletions. From the set of exact and inexact matches we select those with minimum score-based E-values. This gives a set of regions in the reference genome which is aligned to the read using Myers bitvector algorithm [Myers 1999].

Our method utilizes enhanced suffix arrays [Abouelhoda 2004] to quickly find the exact and inexact matches. It maps more reads and achieves higher recall rates than previous methods. This consistently holds for reads produced by 454 as well as Illumina sequencing technologies. The presented a novel read mapping approach presented is able to efficiently handle $3'$ and $5'$ contaminations as well as mismatches, insertions and deletions in short and medium length reads. It is based on a matching model with inexact seeds containing mismatches, insertions and deletions. The sensitivity and specificity of our method is controlled by a maximum seed differences threshold, a maximum occurrence threshold, an E-value threshold and an identity threshold. Compared to previous methods, our approach yields improved recall rates especially for reads containing insertions and deletions. In our Benchmarks we have shown that `segemehl` is a highly sensitive read mapper that has substantial advantages over the most commonly used tools for next generation sequencing read alignments. The advantage of `segemehl` is specifically pronounced for mRNAseq reads that typically come with higher error rates as compared to DNAseq. The price for the gain in sensitivity is an increase in running time: with $k = 1$ our method is approximately ten times slower than `Bowtie`, the fastest program in our comparison. As we used enhanced suffix arrays, matching against a large mammalian genome has to be done chromosome by chromosome when off-the-shelf hardware is used. However, the gain in sensitivity for reads with mismatches and the failure of other methods when dealing with indels may be, depending on the users demands, a reasonable trade off for these shortcomings. Meanwhile, `segemehl` has been evaluated in independent publications (cf. [Caboche 2014, Shang 2014]). In their evaluation of the performance of different aligners on Ion Torrent reads Caboche et al. conclude that `segemehl` shows an outstanding robustness, especially when the number of sequencing errors are high. The average F-measure of `segemehl`, a benchmark for the correctness of the mapping, was the highest as compared to 13 other aligners on simulated reads. In addition the authors tested the mapping performance on real data. They found that only three mappers, namely `segemehl`, SHRiMP2 [David 2011], and Bowtie2 [Langmead 2012b], maintained a high number of normalized found intervals (NFI), a measure for the mapping performance introduced by [Holtgrewe 2011a], even for high error rates. Since its publication `segemehl` has been integrated in a number of pipelines or cloud architectures such as READemption [Förstner 2014] or Excalibur [Leite 2014].

In summary, our method is not limited to a specific technology or read length. Although quality values are not considered yet, the matching strategy can easily be adapted to evaluate low quality bases specifically. This is especially interesting since manufacturers try to increase their read lengths at the cost of higher error rates. Thus, the use of `segemehl` seems to be suitable also for future sequencing data. The increased sensitivity of the presented matching model, along with its ability to handle leading and terminal contaminations is a trade off for the large memory requirements of the enhanced suffix arrays. In the future, compressed index structures like the FM-index [Ferragina 2000] may be a suitable framework to implement our matching model with smaller memory requirements.

The core mapping algorithm described here has been extended to allow the mapping of bisulfite-treated sequence data [Otto 2012]. This protocol is used to identify methylated cytosines of the DNA. The Methyl-groups are basic epigenetic modifications and play a role in the regulation of transcription. The bisulfite treatment converts unmethylated cytosines to uracils while methylated cytosines are protected from this reaction. Since the vast majority of DNA's cytosines are not methylated this treatment *practically* amounts to the collapse of the four letter DNA alphabet $\{A, C, G, T\}$ to a three letter alphabet $\{A, G, T\}$. To enable the seed search on both strands, two reference genomes and their corresponding ESAs, one with C-to-T and one with G-to-A conversions, are created. Since forward (C-to-T) and backward (G-to-A) ESA can be used consecutively, only disk storage but not core memory is affected. The reduced alphabet requires somewhat longer seeds to ensure unambiguous matches, leading to an increase in runtime by a small constant factor compared to ordinary read matching. Subsequently, a modified bitvector algorithm that is aware of the bisulfite conversions is applied. This tool is currently used in the context of a major bisulfite sequencing projects to evaluate the influences of DNA-methylation in lymphomas.

In addition have presented a novel algorithm for split read mapping of single-end RNA-seq data that combines error-tolerant ESA-based seed mapping with a fast bit-vector alignment. It accommodates multiple splits within a single read and makes no *a priori* assumptions on the transcript structure. Implemented in the `segemehl` mapping tool, it readily identifies conventional splice junctions, co-linear, and non-collinear fusion transcripts, and trans-spliced RNAs without the need for separate post-processing or extensive computational overhead. Compared to widely used competitors, the method has significantly higher sensitivity and reduces false positive results in particular in trans-splicing scenarios. This makes `segemehl` the method of choice to annotate rare transcript variants. Indeed, previously undescribed exons and additional splice junctions are readily identified.

Strikingly, the precision is maintained even for reads with higher error rates (Supplementary Figure 2). This features is of particular interest when transcriptome data from organisms with high allelic variations are processed. It also makes it feasible to analyze transcriptome data by mapping to the genome of a closely related species as reference.

Already the analysis of the few test data sets used here to verify the viability of our approach show that RNA-seq data sets readily contain evidence for a substantial number of transcripts with atypical structures. In addition to read-through transcripts, which preserve co-linearity and can be explained by conventional splicing from an extended primary transcript, we also observe a moderate number of products that violate co-linearity. These fall into at least three broad classes: strand-reversing RNAs such as the fruit fly MODMDG4 gene [Dorn 2001, McManus 2010] that originate from both reading directions of a compact locus, permuted RNAs and circular isoforms [Salzman 2012, Salzman 2013, Zhang 2013, Jeck 2013, Hansen 2013, Memczak 2013] as in the case of ANRIL ncRNA [Burd 2010], and RNAs that are combined from components originating from different loci such as the rat HongrES2 RNA [Ni 2011] and several chimeric human proteins recently described in [Frenkel-Morgenstern 2012a]. A few of these cases were studied in some more detail and least in part also characterized functionally [Frenkel-Morgenstern 2012b]. These studies suggest that at least a part of the non-collinear transcriptome [Gingeras 2009] is functional and cannot be explained as a consequence of chromosomal rearrangements relative to the reference genome. An accurate mapping tool such as `segemehl` that is sensitive to split reads and operates without an underlying model of valid transcript structure and hence does not discard non-collinear mapping results as artefacts, therefore is an indispensable tool for a systematic investigations into this largely uncharted section of the transcriptome. Additionally, we have identified a new, potentially tissue specific isoform of the tumor suppressor gene p53.

Finally, we have presented a data adaptive model for variant calling based on easily accessible read characteristics, namely the log-likelihoods of nucleotide qualities, relative read positions, alignment errors, multiple hits and the mismatch rate at a position to obtain a score. With the exception of nucleotide qualities, which are provided as input by the sequencing method, all log-likelihoods are sampled from the data itself. We show that in simulated as well as in real data sets this score gives rise to a mixture distribution that distinguishes between true variants and noise. A spline fit to the overall marginal density allows us to determine a decision threshold that optimally separates these score populations. In the simulated data we demonstrated that this simple model is at par with two of the most commonly used probabilistic models for SNV calling methods in terms of both sensitivity and specificity. When applying our model to actual next-generation sequencing data, we observe that the distributions of the scores vary significantly among the different data sets. As expected, the clearest separation of the mixture was obtained for the haploid *E. coli* data set. In addition, the small size of the genome and the abscence of repetitive elements probably improves the separability of the scores. The situation for the two diploid genomes *A. thaliana* and *D. melanogaster* is different. While both genomes have comparable sizes, the separability of the score distributions varies strongly among these two data sets. While a clear minimum can be found for the plant, the mixture in *D. melanogaster* appears to be more complicated. In this case, by construction our model selects a conservative decision threshold. While the number of calls is similar to the other probabilistic SNV callers in the simulations as well as the next-generations data sets of the plant and the bacteria, it is significantly reduced in the fruit fly data set. These differences indicate that the characteristics of next-generation data sets have a strong impact on the success of variant calling. Furthermore, we observe that, at least for the score proposed here, a significant difference of the separability of the mixture distribution can be found between simulated and real data. Thus, we argue that data adaptive components could help to balance the trade-off between sensitivity and specificity.

The features of our mapping approach have been useful in a number of own projects. Besides the identification of small RNAs in *Helicobacter pylori* and the genome-wide mapping of transcription start sites, segemehl's capabilities for multiple mapping of ChIP-Seq reads revealed the potential interaction of the long non-coding RNA ANRIL with an ALU-DEIN repeat. Meanwhile, our mapping algorithm has been recommended and successfully applied to other next-generation sequencing data sets such as PAR-CLIP [Kishore 2013, König 2012]. This protocol is aimed at studying RNA-protein interaction. After a crosslinking of the RNA and Proteins, the proteins of interest are extracted via immunoprecipitation. After the degradation of the proteins, the RNAs are reverse transcribed and sequenced using a HTS technology.

Thus, `segemehl`'s feature to report multiple alignments has helped to establish the hypothesis that long non-coding RNAs unfold their regulatory potential via repetitive elements. Within the lymphoma project segemehl's split read capabilities have helped to identify mutated isoforms of ID3 and to identify new spliced transcripts. In addition, the fundamental models for automatically annotating transcription start sites from dRNAseq data [Sharma 2010, Schmidtke 2012] have been extended by other researchers and became available to a broader audience in the form of a web server [Amman 2014].

Appendix A

Appendix

Parameters for running split-read benchmarks

For the comparisons, we have used the following tools and options in all tested scenarios.

`GSNAP` (version 2013-11-27) [Wu 2010],

```
>gsnap -A sam --nofails -D GSNAP -d hg19_gsnap -N 1 -t 15 file.fq
```

`MapSplice` (version 2.1.2) [Wang 2010],

```
>python mapsplice.py -1 file.fq -c hg19/singlechroms/ -x hg19/ -Q fq --fusion-non-canonical --non-canonical\\
                     -p 15 --qual-scale phred33
```

`RUM` (version 1.12_01) [Grant 2011],

```
>RUM_runner.pl rum.config_hg19 file.fq outfolder 1 name
```

`segemehl` (version 0.1.5),

```
>segemehl.x -q file.fq -d hg19.fa -i hg19.idx -S -t 15 -s -o file.out
>testrealign.x -d hg19.fa -q file.out -n
```

`SOAPsplice` (version 1.9) [Huang 2011],

```
>soapsplice -d hg19.fa.index -1 file.fq -o file.out -p 15
```

`SpliceMap` (version 3.3.5.2 (55)) [Au 2010],

```
>runSpliceMap SpliceMap.cfg (default, except num_threads = 15)
```

`STAR` (version 2.1.3e_r157) [Dobin 2013b],

```
>STAR --genomeDir hg19_STAR/ --readFilesIn file.fq --runThreadN 15 --genomeLoad NoSharedMemory
```

`TopHat2` (version 2.0.4) [Trapnell 2009, Kim 2011a],

```
>tophat2 -p 15 -o outFolder --fusion-search bowtie/hg19 file.fq
```

For the long RefSeq reads, we used STAR following the supplement of [Dobin 2013b].

`STAR` (version 2.3.1c; compiled as 'STARlong') [Dobin 2013b],

```
>STAR --genomeDir hg19_STAR/ --readFilesIn file.fq --runThreadN 15 --outFilterMismatchNmax 100 \\
      --seedSearchLmax 30 --seedSearchStartLmax 30 --seedPerReadNmax 100000 --seedPerWindowNmax 100\\
      --alignTranscriptsPerReadNmax 100000 --alignTranscriptsPerWindowNmax 10000 \\
      --chimSegmentMin 17 --chimScoreMin 17
```

Parameter for variant simulation and benchmarking

Read simulation

For the simulation or reads and allel contents we used the program GemSIM (v. 1.6). We simulated reads for the human chromosome 21 (hg19) with different coverages using an Illumina specific error model (ill100v5_s)

```
python GemReads.py -g simulatedsnps.txt -r chr21.fasta -m ill100v5_s.gzip \
-n <noofreads> -l 100 -q 64 -o <mysimulatedreads.fq>
```

Next generation sequencing data

Paired end next generation sequencing data for Arabidopsis thaliana (SRR519713), Escherichia coli (ERR163894) and Drosophila melanogaster (SRR1177123) was downloaded under the respecitve accession numbers from the Short Read Archive (`www.ncbi.nlm.nih.gov/sra`).

Benchmarks

For the benchmarks we have aligned the simulated as well as the real reads with bwa and called the variants with SAMtools and GATK. For our own model the reads were aligned using segemehl. The command line parameters and version numbers are given below.

a) BWA v 0.6.2

```
bwa aln <ref.fa>  <reads1.fq> > bwa_PE1.sai
bwa aln <ref.fa> <reads2.fq>  > bwa_PE2.sai
bwa sampe <ref.fa> bwa_PE1 bwa_PE2 <read1.fa> <read2.fa> > my.sam
```

b) GATK v 2.8.1 (GenomeAnalysisTK-2.8-1-g932cd3a)

calling:

```
java -jar GenomeAnalysisTK.jar -T UnifiedGenotyper -R <ref.fa> \
     -I <sorted.bam> -o <calls.vcf>
```

filtering:

```
java -jar GenomeAnalysisTK.jar -T VariantFiltration -R <ref.fa> \
   -V <calls.vcf> --filterExpression \
   "QD < 2.0 || FS > 60.0 || MQ < 40.0 || HaplotypeScore > 13.0 || \
   MQRankSum < -12.5 || ReadPosRankSum < -8.0" --filterName "filter" \
   -o <calls.filtered.vcf>
```

c) SAMtools v 0.1.19

calling:

```
samtools mpileup -uf <ref.fa> <sorted.bam> | \
         bcftools view -bvcg - > <var.raw.bcf>

bcftools view var.raw.bcf > <calls.vcf>
```

filtering:

```
varFilter -D100 > <calls.filter.vcf>
```

d) segemehl v 0.1.7

```
segemehl.x -d <ref.fa> -i <ref.idx> -q SRR519713.fastq -D 0 > mysam.sam
```

Bibliography

[Abouelhoda 2004] M. I. Abouelhoda, S. Kurtz and E. Ohlebusch. *Replacing suffix trees with enhanced suffix arrays*. J. Discrete Algorithms, vol. 2, no. 1, pages 53–86, 2004. (Cited on pages 38, 40, 42, 44, 54, 55 and 121.)

[Allen 2011] M. A. Allen, L. W. Hillier, R. H. Waterston and T. Blumenthal. *A global analysis of C. elegans trans-splicing*. Genome Research, vol. 21, no. 2, pages 255–264, 2011. (Cited on page 83.)

[Alm 1999] R. A. Alm, L. S. Ling, D. T. Moir, B. L. King, E. D. Brown, P. C. Doig, D. R. Smith, B. Noonan, B. C. Guild, B. L. deJonge, G. Carmel, P. J. Tummino, A. Caruso, M. Uria-Nickelsen, D. M. Mills, C. Ives, R. Gibson, D. Merberg, S. D. Mills, Q. Jiang, D. E. Taylor, G. F. Vovis and T. J. Trust. *Genomic-sequence comparison of two unrelated isolates of the human gastric pathogen Helicobacter pylori*. Nature, vol. 397, no. 6715, pages 176–80, Jan 1999. (Cited on page 102.)

[Altschul 1990] S. F. Altschul, W. Gish, W. Miller, E. W. Myers and D. J. Lipman. *Basic local alignment search tool*. Journal of Molecular Biology, vol. 215, no. 3, pages 403 – 410, 1990. (Cited on page 26.)

[Amaral 2008] P. P. Amaral and J. S. Mattick. *Noncoding RNA in development*. Mamm Genome, vol. 19, no. 7-8, pages 454–92, Aug 2008. (Cited on page 119.)

[Ameur 2010] A. Ameur, A. Wetterbom, L. Feuk and U. Gyllensten. *Global and unbiased detection of splice junctions from RNA-seq data*. Genome Biol, vol. 11, no. 3, page R34, 2010. (Cited on page 70.)

[Amman 2014] F. Amman, M. Wolfinger, R. Lorenz, I. Hofacker, P. Stadler and S. FindeiSZ. *TSSAR: TSS annotation regime for dRNA-seq data*. BMC Bioinformatics, vol. 15, no. 1, page 89, 2014. (Cited on page 125.)

[Anderson 1998] J. Anderson, L. Phan, R. Cuesta, B. A. Carlson, M. Pak, K. Asano, G. R. Björk, M. Tamame and A. G. Hinnebusch. *The essential Gcd10p-Gcd14p nuclear complex is required for 1-methyladenosine modification and maturation of initiator methionyl-tRNA*. Genes Dev, vol. 12, no. 23, pages 3650–62, Dec 1998. (Cited on page 114.)

[Au 2010] K. F. Au, H. Jiang, L. Lin, Y. Xing and W. H. Wong. *Detection of splice junctions from paired-end RNA-seq data by SpliceMap*. Nucleic Acids Res, vol. 38, no. 14, pages 4570–8, Aug 2010. (Cited on pages 70 and 127.)

[Baeten 2003] J. C. M. Baeten, J. K. Lenstra, J. Parrow and G. J. Woeginger, editors. Automata, languages and programming, 30th international colloquium, icalp 2003, eindhoven, the netherlands, june 30 - july 4, 2003. proceedings, volume 2719 of *Lecture Notes in Computer Science*. Springer, 2003. (Cited on pages 35 and 140.)

[Baltrus 2009] D. A. Baltrus, M. R. Amieva, A. Covacci, T. M. Lowe, D. S. Merrell, K. M. Ottemann, M. Stein, N. R. Salama and K. Guillemin. *The complete genome sequence of Helicobacter pylori strain G27*. J Bacteriol, vol. 191, no. 1, pages 447–8, Jan 2009. (Cited on page 102.)

[Batzer 2002] M. A. Batzer and P. L. Deininger. *Alu repeats and human genomic diversity*. Nat Rev Genet, vol. 3, no. 5, pages 370–9, May 2002. (Cited on page 119.)

[Benjamini 1995] Y. Benjamini and Y. Hochberg. *Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing*. Journal of the Royal Statistical Society. Series B (Methodological), vol. 57, no. 1, pages 289–300, 1995. (Cited on page 109.)

[Bentley 2008] D. R. Bentley, S. Balasubramanian, H. P. Swerdlow, G. P. Smith, J. Milton, C. G. Brown, K. P. Hall, D. J. Evers, C. L. Barnes, H. R. Bignell, J. M. Boutell, J. Bryant, R. J. Carter, R. Keira Cheetham, A. J. Cox, D. J. Ellis, M. R. Flatbush, N. A. Gormley, S. J. Humphray, L. J. Irving, M. S. Karbelashvili, S. M. Kirk, H. Li, X. Liu, K. S. Maisinger, L. J. Murray, B. Obradovic, T. Ost, M. L. Parkinson, M. R. Pratt, I. M. J. Rasolonjatovo, M. T. Reed, R. Rigatti, C. Rodighiero, M. T. Ross, A. Sabot, S. V. Sankar, A. Scally, G. P. Schroth, M. E. Smith, V. P. Smith, A. Spiridou, P. E. Torrance, S. S. Tzonev, E. H. Vermaas, K. Walter, X. Wu, L. Zhang, M. D. Alam, C. Anastasi, I. C. Aniebo, D. M. D. Bailey, I. R. Bancarz, S. Banerjee, S. G. Barbour, P. A. Baybayan, V. A. Benoit, K. F. Benson, C. Bevis, P. J. Black, A. Boodhun, J. S. Brennan, J. A. Bridgham, R. C. Brown, A. A. Brown, D. H. Buermann, A. A. Bundu, J. C. Burrows, N. P. Carter, N. Castillo, M. Chiara E Catenazzi, S. Chang, R. Neil Cooley, N. R. Crake, O. O. Dada, K. D. Diakoumakos, B. Dominguez-Fernandez, D. J. Earnshaw, U. C. Egbujor, D. W. Elmore, S. S. Etchin, M. R. Ewan, M. Fedurco, L. J. Fraser, K. V. Fuentes Fajardo, W. Scott Furey, D. George, K. J. Gietzen, C. P. Goddard, G. S. Golda, P. A. Granieri, D. E. Green, D. L. Gustafson, N. F. Hansen, K. Harnish, C. D. Haudenschild, N. I. Heyer, M. M. Hims, J. T. Ho, A. M. Horgan, K. Hoschler, S. Hurwitz, D. V. Ivanov,

M. Q. Johnson, T. James, T. A. Huw Jones, G.-D. Kang, T. H. Kerelska, A. D. Kersey, I. Khrebtukova, A. P. Kindwall, Z. Kingsbury, P. I. Kokko-Gonzales, A. Kumar, M. A. Laurent, C. T. Lawley, S. E. Lee, X. Lee, A. K. Liao, J. A. Loch, M. Lok, S. Luo, R. M. Mammen, J. W. Martin, P. G. McCauley, P. McNitt, P. Mehta, K. W. Moon, J. W. Mullens, T. Newington, Z. Ning, B. Ling Ng, S. M. Novo, M. J. O'Neill, M. A. Osborne, A. Osnowski, O. Ostadan, L. L. Paraschos, L. Pickering, A. C. Pike, A. C. Pike, D. Chris Pinkard, D. P. Pliskin, J. Podhasky, V. J. Quijano, C. Raczy, V. H. Rae, S. R. Rawlings, A. Chiva Rodriguez, P. M. Roe, J. Rogers, M. C. Rogert Bacigalupo, N. Romanov, A. Romieu, R. K. Roth, N. J. Rourke, S. T. Ruediger, E. Rusman, R. M. Sanches-Kuiper, M. R. Schenker, J. M. Seoane, R. J. Shaw, M. K. Shiver, S. W. Short, N. L. Sizto, J. P. Sluis, M. A. Smith, J. Ernest Sohna Sohna, E. J. Spence, K. Stevens, N. Sutton, L. Szajkowski, C. L. Tregidgo, G. Turcatti, S. Vandevondele, Y. Verhovsky, S. M. Virk, S. Wakelin, G. C. Walcott, J. Wang, G. J. Worsley, J. Yan, L. Yau, M. Zuerlein, J. Rogers, J. C. Mullikin, M. E. Hurles, N. J. McCooke, J. S. West, F. L. Oaks, P. L. Lundberg, D. Klenerman, R. Durbin and A. J. Smith. *Accurate whole human genome sequencing using reversible terminator chemistry*. Nature, vol. 456, no. 7218, pages 53–9, Nov 2008. (Cited on page 4.)

[Berger 2010] M. F. Berger, J. Z. Levin, K. Vijayendran, A. Sivachenko, X. Adiconis, J. Maguire, L. A. Johnson, J. Robinson, R. G. Verhaak, C. Sougnez, R. C. Onofrio, L. Ziaugra, K. Cibulskis, E. Laine, J. Barretina, W. Winckler, D. E. Fisher, G. Getz, M. Meyerson, D. B. Jaffe, S. B. Gabriel, E. S. Lander, R. Dummer, A. Gnirke, C. Nusbaum and L. A. Garraway. *Integrative analysis of the melanoma transcriptome*. Genome Res, vol. 20, no. 4, pages 413–27, Apr 2010. (Cited on pages 70, 81 and 88.)

[Blumenthal 2005] T. Blumenthal. Wormbook, chapter Trans-splicing and operons. In The C. elegans Research Community [The C. elegans Research Community 2005], 2005. (Cited on page 82.)

[Boerma 2009] E. G. Boerma, R. Siebert, P. M. Kluin and M. Baudis. *Translocations involving 8q24 in Burkitt lymphoma and other malignant lymphomas: a historical review of cytogenetics in the light of todays knowledge*. Leukemia, vol. 23, no. 2, pages 225–34, Feb 2009. (Cited on page 117.)

[Bogerd 2010] H. P. Bogerd, H. W. Karnowski, X. Cai, J. Shin, M. Pohlers and B. R. Cullen. *A mammalian herpesvirus uses noncanonical expression and processing mechanisms to generate viral MicroRNAs*. Mol Cell, vol. 37, no. 1, pages 135–42, Jan 2010. (Cited on page 116.)

[Boll 2012] K. Boll, K. Reiche, K. Kasack, N. Mörbt, A. K. Kretzschmar, J. M. Tomm, G. Verhaegh, J. Schalken, M. von Bergen, F. Horn and J. Hackermüller. *MiR-130a, miR-203 and miR-205 jointly repress key oncogenic pathways and are downregulated in prostate carcinoma*. Oncogene, vol. doi: 10.1038/onc.2012.55, 2012. (Cited on page 86.)

[Brock 2008] J. E. Brock, S. Pourshahian, J. Giliberti, P. A. Limbach and G. R. Janssen. *Ribosomes bind leaderless mRNA in Escherichia coli through recognition of their 5′-terminal AUG*. RNA, vol. 14, no. 10, pages 2159–69, Oct 2008. (Cited on page 109.)

[Brown 2000] L. M. Brown. *Helicobacter pylori: epidemiology and routes of transmission*. Epidemiol Rev, vol. 22, no. 2, pages 283–97, 2000. (Cited on page 101.)

[Bruce 2008] M. G. Bruce and H. I. Maaroos. *Epidemiology of Helicobacter pylori infection*. Helicobacter, vol. 13 Suppl 1, pages 1–6, Oct 2008. (Cited on page 102.)

[Burd 2010] C. E. Burd, W. R. Jeck, Y. Liu, H. K. Sanoff, Z. Wang and N. E. Sharpless. *Expression of Linear and Novel Circular Forms of an INK4/ARF-Associated Non-Coding RNA Correlates with Atherosclerosis Risk*. PLoS Genet., vol. 6, page e1001233, 2010. (Cited on page 123.)

[Bury-Moné 2004] S. Bury-Moné, J.-M. Thiberge, M. Contreras, A. Maitournam, A. Labigne and H. De Reuse. *Responsiveness to acidity via metal ion regulators mediates virulence in the gastric pathogen Helicobacter pylori*. Mol Microbiol, vol. 53, no. 2, pages 623–38, Jul 2004. (Cited on page 111.)

[Caboche 2014] S. Caboche, C. Audebert, Y. Lemoine and D. Hot. *Comparison of mapping algorithms used in high-throughput sequencing: application to Ion Torrent data*. BMC Genomics, vol. 15, no. 1, page 264, 2014. (Cited on page 122.)

[Camus 2012] S. Camus, S. Ménendez, K. Fernandes, N. Kua, G. Liu, D. P. Xirodimas, D. P. Lane and J. C. Bourdon. *The p53 isoforms are differentially modified by Mdm2*. Cell Cycle, vol. 11, pages 1646–1655, 2012. (Cited on page 81.)

[Cavanagh 2010] A. T. Cavanagh, P. Chandrangsu and K. M. Wassarman. *6S RNA regulation of relA alters ppGpp levels in early stationary phase*. Microbiology, vol. 156, no. Pt 12, pages 3791–800, Dec 2010. (Cited on page 107.)

[Chang 1994] W. Chang and E. Lawler. *Sublinear approximate string matching and biological applications.* Algorithmica, vol. 12, pages 327–344, 1994. (Cited on pages 53 and 55.)

[Cho 2002] H. D. Cho, K. Tomita, T. Suzuki and A. M. Weiner. *U2 small nuclear RNA is a substrate for the CCA-adding enzyme (tRNA nucleotidyltransferase)*. J Biol Chem, vol. 277, no. 5, pages 3447–55, Feb 2002. (Cited on page 116.)

[Clarke 2009] J. Clarke, H.-c. Wu, L. Jayasinghe, A. Patel, S. Reid and H. Bayley. *Continuous base identification for single-molecule nanopore DNA sequencing*. Nature Nanotechnology, vol. 4, 2009. (Cited on page 7.)

[Crochemore 2007] M. Crochemore, C. Hancart and T. Lecroq. Algorithms on strings. Cambridge University Press, 2007. (Cited on page 54.)

[Danan 2012] M. Danan, S. Schwartz, S. Edelheit and R. Sorek. *Transcriptome-wide discovery of circular RNAs in Archaea*. Nucleic Acids Res, vol. 40, pages 3131–3142, 2012. (Cited on page 76.)

[David 2011] M. David, M. Dzamba, D. Lister, L. Ilie and M. Brudno. *SHRiMP2: Sensitive yet Practical Short Read Mapping*. Bioinformatics, vol. 27, no. 7, pages 1011–1012, 2011. (Cited on page 122.)

[de Koning 2011] A. P. J. de Koning, W. Gu, T. A. Castoe, M. A. Batzer and D. D. Pollock. *Repetitive Elements May Comprise Over Two-Thirds of the Human Genome*. PLoS Genet, vol. 7, no. 12, page e1002384, 12 2011. (Cited on page 119.)

[DePristo 2011] M. A. DePristo, E. Banks, R. Poplin, K. V. Garimella, J. R. Maguire, C. Hartl, A. A. Philippakis, G. del Angel, M. A. Rivas, M. Hanna, A. McKenna, T. J. Fennell, A. M. Kernytsky, A. Y. Sivachenko, K. Cibulskis, S. B. Gabriel, D. Altshuler and M. J. Daly. *A framework for variation discovery and genotyping using next-generation DNA sequencing data.* Nature genetics, vol. 43, no. 5, pages 491–498, 2011. (Cited on pages 12 and 91.)

[Djebali 2012] S. Djebali, C. A. Davis, A. Merkel, A. Dobin, T. Lassmann, A. M. Mortazavi, A. Tanzer, J. Lagarde, W. Lin, F. Schlesinger, C. Xue, G. K. Marinov, J. Khatun, B. A. Williams, C. Zaleski, J. Rozowsky, M. Röder, F. Kokocinski, R. F. Abdelhamid, T. Alioto, I. Antoshechkin, M. T. Baer, P. Batut, K. Bell, I. Bell, S. Chakrabortty, X. Chen, J. Chrast, J. Curado, T. Derrien, J. Drenkow, E. Dumais, J. Dumais, R. Duttagupta, E. Falconnet, M. Fastuca, K. Fejes-Toth, P. Ferreira, S. Foissac, M. J. Fullwood, H. Gao, D. Gonzalez, A. Gordon, H. Gunawardena, C. Howald, S. Jha, R. Johnson, P. Kapranov, B. King, C. Kingswood, O. J. Luo, E. Park, J. B. Preall, K. Persaud, P. Ribeca, B. Risk, D. Robyr, M. Sammeth, L.-H. See, A. Shahab, L. Schaffer, J. Skancke, A. M. Suzuki, H. Takahashi,

H. Tilgner, D. Trout, N. Walters, H. Wang, J. Wrobel, Y. Yu, X. Ruan, Y. Hayashizaki, J. Harrow, M. Gerstein, T. Hubbard, A. Reymond, S. E. Antonarakis, G. Hannon, M. C. Giddings, Y. Ruan, B. Wold, P. Carninci, R. Guigó and T. R. Gingeras. *Landscape of transcription in human cells*. Nature, vol. in press, 2012. (Cited on pages 81, 85 and 88.)

[Dobin 2013a] A. Dobin, C. Davis, F. Schlesinger, J. Drenkow, C. Zaleski, S. Jha, P. Batut, M. Chaisson and T. Gingeras. *STAR: ultrafast universal RNA-seq aligner*. Bioinformatics, vol. 29, pages 15–21, Jan 2013. (Cited on page 67.)

[Dobin 2013b] A. Dobin, C. A. Davis, F. Schlesinger, J. Drenkow, C. Zaleski, S. Jha, P. Batut, M. Chaisson and T. R. Gingeras. *STAR: ultrafast universal RNA-seq aligner*. Bioinformatics, vol. 29, no. 1, pages 15–21, 2013. (Cited on pages 71 and 127.)

[Dohm 2008] J. C. Dohm, C. Lottaz, T. Borodina and H. Himmelbauer. *Substantial biases in ultra-short read data sets from high-throughput DNA sequencing.* Nucleic acids research, vol. 36, page e105, 2008. (Cited on pages 9 and 48.)

[Dorn 2001] R. Dorn, G. Reuter and A. Loewendorf. *Transgene analysis proves mRNA trans-splicing at the complex mod(mdg4) locus in Drosophila*. Proc Natl Acad Sci U S A, vol. 98, no. 17, pages 9724–9, Aug 2001. (Cited on pages 70, 81 and 123.)

[Drmanac 2009] R. Drmanac, A. B. Sparks, M. J. Callow, A. L. Halpern, N. L. Burns, B. G. Kermani, P. Carnevali, I. Nazarenko, G. B. Nilsen, G. Yeung, F. Dahl, A. Fernandez, B. Staker, K. P. Pant, J. Baccash, A. P. Borcherding, A. Brownley, R. Cedeno, L. Chen, D. Chernikoff, A. Cheung, R. Chirita, B. Curson, J. C. Ebert, C. R. Hacker, R. Hartlage, B. Hauser, S. Huang, Y. Jiang, V. Karpinchyk, M. Koenig, C. Kong, T. Landers, C. Le, J. Liu, C. E. Mcbride, M. Morenzoni, R. E. Morey, K. Mutch, H. Perazich, K. Perry, B. A. Peters, J. Peterson, C. L. Pethiyagoda, K. Pothuraju, C. Richter, A. M. Rosenbaum, S. Roy, J. Shafto, U. Sharanhovich, K. W. Shannon, C. G. Sheppy, M. Sun, J. V. Thakuria, A. Tran, D. Vu, A. W. Zaranek, X. Wu, S. Drmanac, A. R. Oliphant, W. C. Banyai, B. Martin, D. G. Ballinger, G. M. Church and C. A. Reid. *Human Genome Sequencing Using Unchained Base Reads on Self-Assembling DNA Nanoarrays*. Science, pages 1–7, 2009. (Cited on page 7.)

[Duan 2012] S. Duan, L. Cermak, J. K. Pagan, M. Rossi, C. Martinengo, P. F. di Celle, B. Chapuy, M. Shipp, R. Chiarle and M. Pagano. *FBXO11 targets BCL6 for degradation and is inactivated in diffuse large*

B-cell lymphomas. Nature, vol. 481, no. 7379, pages 90–93, Jan 2012. (Cited on page 118.)

[Ebhardt 2009] H. A. Ebhardt, H. H. Tsang, D. C. Dai, Y. Liu, B. Bostan and R. P. Fahlman. *Meta-analysis of small RNA-sequencing errors reveals ubiquitous post-transcriptional RNA modifications*. Nucleic Acids Res, vol. 37, no. 8, pages 2461–70, May 2009. (Cited on page 114.)

[Edgren 2011] H. Edgren, A. Murumagi, S. Kangaspeska, D. Nicorici, V. Hongisto, K. Kleivi, I. H. Rye, S. Nyberg, M. Wolf, A.-L. Borresen-Dale and O. Kallioniemi. *Identification of fusion genes in breast cancer by paired-end RNA-sequencing*. Genome Biol, vol. 12, no. 1, page R6, Jan 2011. (Cited on page 70.)

[Efron 1996] B. Efron and R. Tibshirani. *Using specially designed exponential families for density estimation*. Annals Stat., vol. 24, pages 2431–2461, 1996. (Cited on page 95.)

[Eid 2009] J. Eid, A. Fehr, J. Gray, K. Luong, J. Lyle, G. Otto, P. Peluso, D. Rank, P. Baybayan, B. Bettman, A. Bibillo, K. Bjornson, B. Chaudhuri, F. Christians, R. Cicero, S. Clark, R. Dalal, A. Dewinter, J. Dixon, M. Foquet, A. Gaertner, P. Hardenbol, C. Heiner, K. Hester, D. Holden, G. Kearns, X. Kong, R. Kuse, Y. Lacroix, S. Lin, P. Lundquist, C. Ma, P. Marks, M. Maxham, D. Murphy, I. Park, T. Pham, M. Phillips, J. Roy, R. Sebra, G. Shen, J. Sorenson, A. Tomaney, K. Travers, M. Trulson, J. Vieceli, J. Wegener, D. Wu, A. Yang, D. Zaccarin, P. Zhao, F. Zhong, J. Korlach and S. Turner. *Real-time DNA sequencing from single polymerase molecules*. Science, vol. 323, no. 5910, pages 133–8, Jan 2009. (Cited on page 7.)

[English 2012] A. C. English, S. Richards, Y. Han, M. Wang, V. Vee, J. Qu, X. Qin, D. M. Muzny, J. G. Reid, K. C. Worley and R. A. Gibbs. *Mind the gap: upgrading genomes with Pacific Biosciences RS long-read sequencing technology.* PLoS One, vol. 7, no. 11, 2012. (Cited on page 7.)

[Erlich 2008] Y. Erlich, P. P. Mitra, M. delaBastide, W. R. McCombie and G. J. Hannon. *Alta-Cyclic: a self-optimizing base caller for next-generation sequencing*. Nat Methods, vol. 5, no. 8, pages 679–82, Aug 2008. (Cited on page 8.)

[Ferragina 2000] P. Ferragina and G. Manzini. *Opportunistic data structures with applications*. Proceedings 41st Annual Symposium on Foundations of Computer Science, pages 390–398, 2000. (Cited on pages 10 and 122.)

[Findeiss 2011] S. Findeiss, D. Langenberger, P. F. Stadler and S. Hoffmann. *Traces of post-transcriptional RNA modifications in deep sequencing data*. Biol Chem, vol. 392, no. 4, pages 305–13, Apr 2011. (Cited on pages 101 and 111.)

[Fischer 2007] J. Fischer and V. Heun. *A new succinct representation of RMQ-information and improvements in the enhanced suffix array*. In PROC. ESCAPE. LNCS, pages 459–470. Springer, 2007. (Cited on page 42.)

[Fischer 2010] J. Fischer and V. Heun. *Finding Range Minima in the Middle: Approximations and Applications*. Mathematics in Computer Science, vol. 3, no. 1, pages 17–30, March 2010. (Cited on page 44.)

[Fonseca 2012] N. A. Fonseca, J. Rung, A. Brazma and J. C. Marioni. *Tools for mapping high-throughput sequencing data*. Bioinformatics, vol. 28, no. 24, pages 3169–3177, 2012. (Cited on pages 10 and 47.)

[Förstner 2014] K. U. Förstner, J. Vogel and C. M. M. Sharma. *READemption-A tool for the computational analysis of deep-sequencing-based transcriptome data*. bioRxiv, 2014. (Cited on page 122.)

[Frenkel-Morgenstern 2012a] M. Frenkel-Morgenstern and A. Valencia. *Novel domain combinations in proteins encoded by chimeric transcripts*. Bioinformatics, vol. 28, pages i67–i74, 2012. (Cited on page 123.)

[Frenkel-Morgenstern 2012b] M. F.-M. Frenkel-Morgenstern, V. Lacroix, I. Ezkurdia, Y. Levin, A. Gabashvili, J. Prilusky, A. del Pozo, M. Tress, R. Johnson, R. Guigó and A. Valencia. *Chimeras taking shape: Potential functions of proteins encoded by chimeric RNA transcripts*. Genome Res., vol. doi: 10.1101/gr.130062.111, 2012. (Cited on pages 70 and 123.)

[GenomeWeb 2012] GenomeWeb. *Helicos BioSciences Files for Chapter 11 Bankruptcy Protection*. `http://www.genomeweb.com/sequencing/helicos-biosciences-files-chapter-11-bankruptcy-protection`, 2012. Accessed 2014-04-30. (Cited on pages 3 and 6.)

[GenomeWeb 2013] GenomeWeb. *Roche Shutting Down 454 Sequencing Business*. `http://www.genomeweb.com/sequencing/roche-shutting-down-454-sequencing-business`, 2013. Accessed 2014-04-30. (Cited on page 4.)

[Giegerich 1997] R. Giegerich and S. Kurtz. *From Ukkonen to McCreight and Weiner: A Unifying View of Linear-Time Suffix Tree Construction*. Algorithmica, vol. 19, no. 3, pages 331–353, 1997. (Cited on pages 32 and 33.)

[Gingeras 2009] T. R. Gingeras. *Implications of chimaeric non-co-linear transcripts*. Nature, vol. 461, pages 206–211, 2009. (Cited on page 123.)

[Gonnet 1992] G. H. Gonnet, R. A. Baeza-Yates and T. Snider. *New Indices for Text: Pat Trees and Pat Arrays*. In Information Retrieval: Data Structures & Algorithms, pages 66–82. 1992. (Cited on page 34.)

[Grant 2011] G. R. Grant, M. H. Farkas, A. D. Pizarro, N. F. Lahens, J. Schug, B. P. Brunk, C. J. Stoeckert, J. B. Hogenesch and E. A. Pierce. *Comparative analysis of RNA-Seq alignment algorithms and the RNA-Seq unified mapper (RUM)*. Bioinformatics, vol. 27, no. 18, pages 2518–2528, 2011. (Cited on pages 71 and 127.)

[Graveley 2010] B. R. Graveley, A. N. Brooks, J. W. Carlson, M. O. Duff, J. M. Landolin, L. Yang, C. G. Artieri, M. J. van Baren, N. Boley, B. W. Booth*et al.* *The developmental transcriptome of Drosophila melanogaster*. Nature, vol. 471, no. 7339, pages 473–479, 2010. (Cited on page 88.)

[Hansen 2013] T. B. Hansen, T. I. Jensen, B. H. Clausen, J. B. Bramsen, B. Finsen, C. K. Damgaard and J. Kjems. *Natural RNA circles function as efficient microRNA sponges*. Nature, vol. 495, pages 384–388, 2013. (Cited on pages 70 and 123.)

[Harris 2009] T. D. e. a. Harris. *Single-Molecule DNA Sequencing of a Viral Genome*. Science, vol. 106, 2009. (Cited on pages 5 and 13.)

[Hartmann 2009] R. K. Hartmann, M. Gössringer, B. Späth, S. Fischer and A. Marchfelder. *The making of tRNAs and more - RNase P and tRNase Z*. Prog Mol Biol Transl Sci, vol. 85, pages 319–68, 2009. (Cited on page 115.)

[Hazkani-Covo 2010] E. Hazkani-Covo, R. M. Zeller and W. Martin. *Molecular poltergeists: mitochondrial DNA copies (numts) in sequenced nuclear genomes*. PLoS Genet, vol. 6, no. 2, page e1000834, Feb 2010. (Cited on page 113.)

[Helgadottir 2007] A. Helgadottir, G. Thorleifsson, A. Manolescu, S. Gretarsdottir, T. Blondal, A. Jonasdottir, A. Jonasdottir, A. Sigurdsson, A. Baker, A. Palsson, G. Masson, D. F. Gudbjartsson, K. P. Magnusson, K. Andersen, A. I. Levey, V. M. Backman, S. Matthiasdottir, T. Jonsdottir, S. Palsson, H. Einarsdottir, S. Gunnarsdottir, A. Gylfason, V. Vaccarino, W. C. Hooper, M. P. Reilly, C. B. Granger, H. Austin, D. J. Rader, S. H. Shah, A. A. Quyyumi, J. R. Gulcher, G. Thorgeirsson, U. Thorsteinsdottir, A. Kong and K. Stefansson. *A common variant on chromosome 9p21 affects the risk of myocardial infarction*. Science, vol. 316, no. 5830, pages 1491–3, Jun 2007. (Cited on page 119.)

[Hillier 2009] L. W. Hillier, V. Reinke, P. Green, M. Hirst, M. A. Marra and R. H. Waterston. *Massively parallel sequencing of the polyadenylated transcriptome of C. elegans*. Genome Research, vol. 19, no. 4, pages 657–666, 2009. (Cited on page 83.)

[Hoffmann 2009] S. Hoffmann, C. Otto, S. Kurtz, C. M. Sharma, P. Khaitovich, J. Vogel, P. F. Stadler and J. Hackermüller. *Fast mapping of short sequences with mismatches, insertions and deletions using index structures.* PLoS computational biology, vol. 5, page e1000502, 2009. (Cited on pages 11, 29, 47, 71, 75, 82 and 96.)

[Hoffmann 2011] S. Hoffmann. *Computational analysis of high throughput sequencing data*. Methods Mol Biol, vol. 719, pages 199–217, 2011. (Cited on page 1.)

[Hoffmann 2013] M. Hoffmann, T. Muruvanda, M. W. Allard, J. Korlach, R. J. Roberts, R. Timme, J. Payne, P. F. McDermott, P. Evans, J. Meng, E. W. Brown and S. Zhao. *Complete Genome Sequence of a Multidrug-Resistant Salmonella enterica Serovar Typhimurium var. 5- Strain Isolated from Chicken Breast*. Genome Announcements, vol. 1, no. 6, 2013. (Cited on page 7.)

[Hoffmann 2014a] S. Hoffmann, C. Otto, G. Doose, A. Tanzer, D. Langenberger, S. Christ, M. Kunz, L. Holdt, D. Teupser, J. Hackermüller and P. Stadler. *A multi-split mapping algorithm for circular RNA, splicing, trans-splicing and fusion detection*. Genome Biology, vol. 15, no. 2, page R34, 2014. (Cited on pages 11, 69, 82 and 86.)

[Hoffmann 2014b] S. Hoffmann, P. F. Stadler and K. Strimmer. *A Simple Data-Adaptive Probabilistic Variant Calling Model.* in preparation, 2014. (Cited on page 89.)

[Holdt 2013] L. M. Holdt, S. Hoffmann, K. Sass, D. Langenberger, M. Scholz, K. Krohn, K. Finstermeier, A. Stahringer, W. Wilfert, F. Beutner, S. Gielen, G. Schuler, G. Gäbel, H. Bergert, I. Bechmann, P. F. Stadler, J. Thiery and D. Teupser. *Alu Elements in ANRIL Non-Coding RNA at Chromosome 9p21 Modulate Atherogenic Cell Functions through Trans-Regulation of Gene Networks*. PLoS Genet, vol. 9, no. 7, page e1003588, 07 2013. (Cited on pages 86 and 119.)

[Holtgrewe 2011a] M. Holtgrewe, A. K. Emde, D. Weese and K. Reinert. *A novel and well-defined benchmarking method for second generation read mapping*. BMC Bioinformatics, vol. 12, no. 1, pages 210+, 2011. (Cited on page 122.)

[Holtgrewe 2011b] M. Holtgrewe, A.-K. Emde, D. Weese and K. Reinert. *A novel and well-defined benchmarking method for second generation read* *mapping*. BMC Bioinformatics, vol. 12, no. 1, page 210, 2011. (Cited on pages 67 and 74.)

[Hopcroft 2006] J. E. Hopcroft, R. Motwani and J. D. Ullman. Introduction to automata theory, languages, and computation (3rd edition). Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2006. (Cited on page 16.)

[Huang 2011] S. Huang, J. Zhang, R. Li, W. Zhang, Z. He, T.-W. Lam, Z. Peng and S.-M. Yiu. *SOAPsplice: genome-wide ab initio detection of splice junctions from RNA-Seq data*. Frontiers in Genetics, vol. 2, no. 46, 2011. (Cited on pages 71 and 127.)

[Huddleston 2014] J. Huddleston, S. Ranade, M. Malig, F. Antonacci, M. Chaisson, L. Hon, P. H. Sudmant, T. A. Graves, C. Alkan, M. Y. Dennis, R. K. Wilson, S. W. Turner, J. Korlach and E. E. Eichler. *Reconstructing complex regions of genomes using long-read sequencing technology*. Genome Research, 2014. (Cited on page 7.)

[Huse 2007] S. Huse, J. Huber, H. Morrison, M. Sogin and D. Welch. *Accuracy and quality of massively parallel DNA pyrosequencing*. Genome Biology, vol. 8, no. 7, page R143, 2007. (Cited on page 48.)

[Hyyrö 2004] H. Hyyrö. *A Note on Bit-Parallel Alignment Computation*. In M. Simánek and J. Holub, editors, Stringology, pages 79–87. Department of Computer Science and Engineering, Faculty of Electrical Engineering, Czech Technical University, 2004. (Cited on page 25.)

[Iida 2009] K. Iida, H. Jin and J.-K. Zhu. *Bioinformatics analysis suggests base modifications of tRNAs and miRNAs in Arabidopsis thaliana*. BMC Genomics, vol. 10, page 155, 2009. (Cited on page 114.)

[Illumina Inc. 2014] Illumina Inc. *Population power. Extreme throughput. $ 1000 human genome.* `http://www.illumina.com/systems/hiseq-x-sequencing-system.ilmn`, 2014. Accessed 2014-04-30. (Cited on page 2.)

[Jeck 2013] W. R. Jeck, J. A. Sorrentino, K. Wang, M. K. Slevin, C. E. Burd, J. Liu, W. F. Marzluff and N. E. Sharpless. *Circular RNAs are abundant, conserved, and associated with ALU repeats*. RNA, vol. 19, pages 141–157, 2013. (Cited on pages 70 and 123.)

[Johnston 1992] J. M. Johnston and W. L. Carroll. *c-myc hypermutation in Burkitt's lymphoma*. Leuk Lymphoma, vol. 8, no. 6, pages 431–9, Dec 1992. (Cited on page 118.)

[Kärkkäinen 2003] J. Kärkkäinen and P. Sanders. *Simple Linear Work Suffix Array Construction*. In Baeten et al. [Baeten 2003], pages 943–955. (Cited on page 35.)

[Karlin 1990] S. Karlin and S. F. Altschul. *Methods for assessing the statistical significance of molecular sequences features by using general scoring schemes.* Proc. Natl. Acad. Sci. USA, vol. 87, pages 2264–2268, Mar 1990. (Cited on pages 52 and 58.)

[Kasai 2001] T. Kasai, G. Lee, H. Arimura, S. Arikawa and K. Park. *Linear-Time Longest-Common-Prefix Computation in Suffix Arrays and Its Applications*. In Proc. Annual Symposium on Combinatorial Pattern Matching, volume 2089 of *Lecture Notes in Computer Science*, pages 181–192, 2001. (Cited on page 40.)

[Kent 2002] W. J. Kent. *BLAT-The BLAST-Like Alignment Tool*. Genome Research, vol. 12, no. 4, pages 656–664, 2002. (Cited on page 71.)

[Kim 2004] N. Kim, E. A. Marcus, Y. Wen, D. L. Weeks, D. R. Scott, H. C. Jung, I. S. Song and G. Sachs. *Genes of Helicobacter pylori regulated by attachment to AGS cells*. Infect Immun, vol. 72, no. 4, pages 2358–68, Apr 2004. (Cited on page 111.)

[Kim 2011a] D. Kim and S. Salzberg. *TopHat-Fusion: an algorithm for discovery of novel fusion transcripts*. Genome Biology, vol. 12, pages R72+, 2011. (Cited on pages 70 and 127.)

[Kim 2011b] S.-J. Kim, H. Choi, S.-S. Park, C. Chang and E. Kim. *Stearoyl CoA desaturase (SCD) facilitates proliferation of prostate cancer cells through enhancement of androgen receptor transactivation*. Mol Cells, vol. 31, pages 371–377, 2011. (Cited on pages 82 and 87.)

[Kircher 2009] M. Kircher, U. Stenzel and J. Kelso. *Improved base calling for the Illumina Genome Analyzer using machine learning strategies.* Genome biology, vol. 10, page R83, January 2009. (Cited on pages 8 and 9.)

[Kishore 2013] S. Kishore, A. R. Gruber, D. J. Jedlinski, A. P. Syed, H. Jorjani and M. Zavolan. *Insights into snoRNA biogenesis and processing from PAR-CLIP of snoRNA core proteins and small RNA sequencing*. Genome biology, vol. 14, no. 5, page R45, 2013. (Cited on page 124.)

[Ko 2005] P. Ko and S. Aluru. *Space efficient linear time construction of suffix arrays*. J. Discrete Algorithms, vol. 3, no. 2-4, pages 143–156, 2005. (Cited on pages 35 and 36.)

[König 2012] J. König, K. Zarnack, N. M. Luscombe and J. Ule. *Protein–RNA interactions: new genomic technologies and perspectives*. Nature

Reviews Genetics, vol. 13, no. 2, pages 77–83, 2012. (Cited on page 124.)

[Koscielny 2009] G. Koscielny, V. Le Texier, C. Gopalakrishnan, V. Kumanduri, J.-J. Riethoven, F. Nardone, E. Stanley, C. Fallsehr, O. Hofmann, M. Kull, E. Harrington, S. Boué, E. Eyras, M. Plass, F. Lopez, W. Ritchie, V. Moucadel, T. Ara, H. Pospisil, A. Herrmann, J. G Reich, R. Guigó, P. Bork, M. v. K. Doeberitz, J. Vilo, W. Hide, R. Apweiler, T. A. Thanaraj and D. Gautheret. *ASTD: The Alternative Splicing and Transcript Diversity database*. Genomics, vol. 93, no. 3, pages 213–20, Mar 2009. (Cited on page 74.)

[Kotake 2011] Y. Kotake, T. Nakagawa, K. Kitagawa, S. Suzuki, N. Liu, M. Kitagawa and Y. Xiong. *Long non-coding RNA ANRIL is required for the PRC2 recruitment to and silencing of p15(INK4B) tumor suppressor gene*. Oncogene, vol. 30, no. 16, pages 1956–62, Apr 2011. (Cited on page 119.)

[Kurtz 2006] S. Kurtz. Foundations of sequence analysis. Lecture notes. University of Hamburg., 2006. (Cited on page 17.)

[Kusters 2006] J. G. Kusters, A. H. M. van Vliet and E. J. Kuipers. *Pathogenesis of Helicobacter pylori infection*. Clin Microbiol Rev, vol. 19, no. 3, pages 449–90, Jul 2006. (Cited on page 102.)

[Langenberger 2009] D. Langenberger, C. Bermudez-Santana, J. Hertel, S. Hoffmann, P. Khaitovich and P. F. Stadler. *Evidence for human microRNA-offset RNAs in small RNA sequencing data*. Bioinformatics, vol. 25, no. 18, pages 2298–301, Sep 2009. (Cited on page 113.)

[Langmead 2009] B. Langmead, C. Trapnell, M. Pop and S. L. Salzberg. *Ultrafast and memory-efficient alignment of short DNA sequences to the human genome*. Genome Biol, vol. 10, no. 3, page R25, 2009. (Cited on pages 11, 48, 64 and 71.)

[Langmead 2012a] B. Langmead and S. Salzberg. *Fast gapped-read alignment with Bowtie 2*. Nature Methods, vol. 9, pages 357–359, 2012. (Cited on pages 48 and 74.)

[Langmead 2012b] B. Langmead and S. Salzberg. *Fast gapped-read alignment with Bowtie 2*. Nat Methods, vol. 9, 2012. (Cited on pages 67 and 122.)

[Leite 2014] A. F. Leite, T. Raiol, C. Tadonki, M. E. M. Walter, C. Eisenbeis and A. C. M. A. de Melo. *Excalibur: an autonomic cloud architecture for executing parallel applications*. In Proceedings of the Fourth International Workshop on Cloud Data and Platforms, page 2. ACM, 2014. (Cited on page 122.)

[Levenshtein 1966] V. Levenshtein. *Binary Codes Capable of Correcting Deletions, Insertions and Reversals*. Soviet Physics Doklady, vol. 10, page 707, 1966. (Cited on page 17.)

[Li 2008a] H. Li, J. Ruan and R. Durbin. *Mapping short DNA sequencing reads and calling variants using mapping quality scores.* Genome research, vol. 18, pages 1851–8, 2008. (Cited on pages 10, 12, 48 and 64.)

[Li 2008b] R. Li, Y. Li, K. Kristiansen and J. Wang. *`SOAP`: short oligonucleotide alignment program*. Bioinformatics, vol. 24, pages 713–714, 2008. (Cited on pages 48 and 64.)

[Li 2009a] H. Li and R. Durbin. *Fast and accurate short read alignment with Burrows-Wheeler transform.* Bioinformatics (Oxford, England), vol. 25, pages 1754–60, 2009. (Cited on pages 10, 64, 67 and 96.)

[Li 2009b] H. Li, B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis, R. Durbin and . G. P. D. P. Subgroup. *The Sequence Alignment/Map format and SAMtools*. Bioinformatics, vol. 25, no. 16, pages 2078–2079, 2009. (Cited on pages 12, 90 and 96.)

[Li 2009c] R. Li, Y. Li, X. Fang, H. Yang, J. Wang, K. Kristiansen and J. Wang. *SNP detection for massively parallel whole-genome resequencing.* Genome research, vol. 19, pages 1124–32, 2009. (Cited on page 12.)

[Li 2009d] R. Li, C. Yu, Y. Li, T.-W. Lam, S.-M. Yiu, K. Kristiansen and J. Wang. *SOAP2: an improved ultrafast tool for short read alignment.* Bioinformatics (Oxford, England), vol. 25, pages 1966–7, 2009. (Cited on page 11.)

[Li 2010a] H. Li and R. Durbin. *Fast and accurate long-read alignment with Burrows-Wheeler transform*. Bioinformatics, vol. 26, pages 589–95, Mar 2010. (Cited on page 67.)

[Li 2010b] R. Li, W. Fan, G. Tian, H. Zhu, L. He, J. Cai, Q. Huang, Q. Cai, B. Li, Y. Bai, Z. Zhang, Y. Zhang, W. Wang, J. Li, F. Wei, H. Li, M. Jian, J. Li, Z. Zhang, R. Nielsen, D. Li, W. Gu, Z. Yang, Z. Xuan, O. A. Ryder, F. C.-C. Leung, Y. Zhou, J. Cao, X. Sun, Y. Fu, X. Fang, X. Guo, B. Wang, R. Hou, F. Shen, B. Mu, P. Ni, R. Lin, W. Qian, G. Wang, C. Yu, W. Nie, J. Wang, Z. Wu, H. Liang, J. Min, Q. Wu, S. Cheng, J. Ruan, M. Wang, Z. Shi, M. Wen, B. Liu, X. Ren, H. Zheng, D. Dong, K. Cook, G. Shan, H. Zhang, C. Kosiol, X. Xie, Z. Lu, H. Zheng, Y. Li, C. C. Steiner, T. T.-Y. Lam, S. Lin, Q. Zhang, G. Li, J. Tian, T. Gong, H. Liu, D. Zhang, L. Fang, C. Ye, J. Zhang, W. Hu, A. Xu, Y. Ren, G. Zhang, M. W. Bruford, Q. Li, L. Ma,

Y. Guo, N. An, Y. Hu, Y. Zheng, Y. Shi, Z. Li, Q. Liu, Y. Chen, J. Zhao, N. Qu, S. Zhao, F. Tian, X. Wang, H. Wang, L. Xu, X. Liu, T. Vinar, Y. Wang, T.-W. Lam, S.-M. Yiu, S. Liu, H. Zhang, D. Li, Y. Huang, X. Wang, G. Yang, Z. Jiang, J. Wang, N. Qin, L. Li, J. Li, L. Bolund, K. Kristiansen, G. K.-S. Wong, M. Olson, X. Zhang, S. Li, H. Yang, J. Wang and J. Wang. *The sequence and de novo assembly of the giant panda genome*. Nature, vol. 463, no. 7279, pages 311–7, Jan 2010. (Cited on page 12.)

[Li 2010c] R. Li, H. Zhu, J. Ruan, W. Qian, X. Fang, Z. Shi, Y. Li, S. Li, G. Shan, K. Kristiansen, S. Li, H. Yang, J. Wang and J. Wang. *De novo assembly of human genomes with massively parallel short read sequencing*. Genome Res, vol. 20, no. 2, pages 265–72, Feb 2010. (Cited on page 12.)

[Li 2011] H. Li. *A statistical framework for SNP calling, mutation discovery, association mapping and population genetical parameter estimation from sequencing data*. Bioinformatics, vol. 27, no. 21, pages 2987–2993, 2011. (Cited on page 90.)

[Li 2013] H. Li. *Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM*. arXiv preprint arXiv:1303.3997, 2013. (Cited on pages 48 and 67.)

[Li 2014] H. Li. *Towards Better Understanding of Artifacts in Variant Calling from High-Coverage Samples*. arXiv preprint arXiv:1404.0929, 2014. (Cited on page 13.)

[Life technologies 2012] Life technologies. *5500 W Series Genetic Analyzers*. `http://tools.lifetechnologies.com/content/sfs/brochures/5500-w-series-spec-sheet.pdf`, 2012. Accessed 2014-04-30. (Cited on page 7.)

[Lin 2008a] H. Lin, Z. Zhang, M. Q. Zhang, B. Ma and M. Li. *ZOOM! Zillions of oligos mapped*. Bioinformatics, vol. 24, no. 21, pages 2431–2437, 2008. (Cited on page 48.)

[Lin 2008b] L. Lin, S. Shen, A. Tye, J. J. Cai, P. Jiang, B. L. Davidson and Y. Xing. *Diverse splicing patterns of exonized Alu elements in human tissues*. PLoS Genetics, vol. 4, no. 10, page e1000225, 2008. (Cited on page 72.)

[Liu 2013] X. Liu, S. Han, Z. Wang, J. Gelernter and B.-Z. Yang. *Variant Callers for Next-Generation Sequencing Data: A Comparison Study*. PLoS ONE, vol. 8, no. 9, pages e75619+, September 2013. (Cited on page 90.)

[Lynch 2009] M. Lynch. *Estimation of allele frequencies from high-coverage genome-sequencing projects*. Genetics, vol. 182, no. 1, pages 295–301, May 2009. (Cited on page 13.)

[Maaß 2007] M. G. Maaß. *Computing suffix links for suffix trees and arrays*. Inf. Process. Lett., vol. 101, no. 6, pages 250–254, 2007. (Cited on page 45.)

[Manber 1993] U. Manber and G. Myers. *Suffix arrays: a new method for on-line string searches*. SIAM J. Comput., vol. 22, pages 935–948, October 1993. (Cited on pages 34 and 35.)

[Marcel 2011] V. Marcel, M. Olivier, B. Mollereau, P. Hainaut and J.-C. Bourdon. *First International p53 Isoforms Meeting: "p53 isoforms through evolution: from identification to biological function"*. Cell Death Different., vol. 18, pages 563–564, 2011. (Cited on page 81.)

[Marco-Sola 2012] S. Marco-Sola, M. Sammeth, R. Guigó and P. Ribeca. *The GEM mapper: fast, accurate and versatile alignment by filtration*. Nat Methods, vol. 9, pages 1185–8, Dec 2012. (Cited on page 67.)

[Margulies 2005] M. Margulies, M. Egholm, W. E. Altman, S. Attiya, J. S. Bader, L. A. Bemben, J. Berka, M. S. Braverman, Y.-J. Chen, Z. Chen, S. B. Dewell, L. Du, J. M. Fierro, X. V. Gomes, B. C. Godwin, W. He, S. Helgesen, C. H. Ho, C. H. Ho, G. P. Irzyk, S. C. Jando, M. L. I. Alenquer, T. P. Jarvie, K. B. Jirage, J.-B. Kim, J. R. Knight, J. R. Lanza, J. H. Leamon, S. M. Lefkowitz, M. Lei, J. Li, K. L. Lohman, H. Lu, V. B. Makhijani, K. E. McDade, M. P. McKenna, E. W. Myers, E. Nickerson, J. R. Nobile, R. Plant, B. P. Puc, M. T. Ronan, G. T. Roth, G. J. Sarkis, J. F. Simons, J. W. Simpson, M. Srinivasan, K. R. Tartaro, A. Tomasz, K. A. Vogt, G. A. Volkmer, S. H. Wang, Y. Wang, M. P. Weiner, P. Yu, R. F. Begley and J. M. Rothberg. *Genome sequencing in microfabricated high-density picolitre reactors*. Nature, vol. 437, no. 7057, pages 376–80, Sep 2005. (Cited on pages 3 and 6.)

[Marshall 1985] B. J. Marshall, J. A. Armstrong, D. B. McGechie and R. J. Glancy. *Attempt to fulfil Koch's postulates for pyloric Campylobacter*. Med J Aust, vol. 142, no. 8, pages 436–9, Apr 1985. (Cited on page 102.)

[Masek 1980] W. J. Masek and M. S. Paterson. *A faster algorithm computing string edit distances*. Journal of Computer and System Sciences, vol. 20, no. 1, pages 18–31, 1980. (Cited on page 22.)

[McCreight 1976] E. M. McCreight. *A Space-Economical Suffix Tree Construction Algorithm*. J. ACM, vol. 23, pages 262–272, April 1976. (Cited on page 32.)

[McElroy 2012] K. E. McElroy, F. Luciani and T. Thomas. *GemSIM: general, error-model based simulator of next-generation sequencing data*. BMC Genomics, vol. 13, page 74, 2012. (Cited on page 96.)

[McKenna 2010] A. McKenna, M. Hanna, E. Banks, A. Sivachenko, K. Cibulskis, A. Kernytsky, K. Garimella, D. Altshuler, S. Gabriel, M. Daly and M. A. DePristo. *The Genome Analysis Toolkit: A MapReduce framework for analyzing next-generation DNA sequencing data*. Genome Research, vol. 20, no. 9, pages 1297–1303, September 2010. (Cited on pages 12, 90 and 96.)

[McManus 2010] C. J. McManus, M. O. Duff, J. Eipper-Mains and B. R. Graveley. *Global analysis of trans-splicing in* Drosophila. Proc. Natl. Acad. Sci. USA, vol. 107, pages 12975–12979, 2010. (Cited on page 123.)

[McPherson 2007] R. McPherson, A. Pertsemlidis, N. Kavaslar, A. Stewart, R. Roberts, D. R. Cox, D. A. Hinds, L. A. Pennacchio, A. Tybjaerg-Hansen, A. R. Folsom, E. Boerwinkle, H. H. Hobbs and J. C. Cohen. *A common allele on chromosome 9 associated with coronary heart disease*. Science, vol. 316, no. 5830, pages 1488–91, Jun 2007. (Cited on page 119.)

[McPherson 2011] A. McPherson, F. Hormozdiari, A. Zayed, R. Giuliany, G. Ha, M. G. F. Sun, M. Griffith, A. Heravi Moussavi, J. Senz, N. Melnyk, M. Pacheco, M. A. Marra, M. Hirst, T. O. Nielsen, S. C. Sahinalp, D. Huntsman and S. P. Shah. *deFuse: an algorithm for gene fusion discovery in tumor RNA-Seq data*. PLoS Comput Biol, vol. 7, no. 5, page e1001138, May 2011. (Cited on pages 71 and 73.)

[Memczak 2013] S. Memczak, M. Jens, A. Elefsinioti, F. Torti, J. Krueger, A. Rybak, L. Maier, S. D. Mackowiak, L. H. Gregersen, M. Munschauer, A. Loewer, U. Ziebold, M. Landthaler, C. Kocks, F. le Noble and N. Rajewsky. *Circular RNAs are a large class of animal RNAs with regulatory potency*. Nature, vol. 495, pages 333–338, 2013. (Cited on pages 70, 81, 82 and 123.)

[Mercer 2012] T. R. Mercer, D. J. Gerhardt, M. E. Dinger, J. Crawford, C. Trapnell, J. A. Jeddeloh, J. S. Mattick and J. L. Rinn. *Targeted RNA sequencing reveals the deep complexity of the human transcriptome*. Nature biotechnology, vol. 30, pages 99–104, 2012. (Cited on pages 81 and 85.)

[Morris 1987] A. Morris and G. Nicholson. *Ingestion of Campylobacter pyloridis causes gastritis and raised fasting gastric pH*. Am J Gastroenterol, vol. 82, no. 3, pages 192–9, Mar 1987. (Cited on page 102.)

[Myers 1999] G. Myers. *A fast bit-vector algorithm for approximate string matching based on dynamic programming*. J. ACM, vol. 46, pages 395–415, May 1999. (Cited on pages 22, 52, 53, 58, 75 and 121.)

[Needleman 1970] S. B. Needleman and C. D. Wunsch. *A general method applicable to the search for similarities in the amino acid sequence of two proteins*. Journal of Molecular Biology, vol. 48, no. 3, pages 443 – 453, 1970. (Cited on page 19.)

[Ni 2011] M.-J. Ni, Z.-H. Hu, Q. Liu, M.-F. Liu, M.-h. Lu, J.-S. Zhang, L. Zhang and Z. Yong-Lian. *Identification and Characterization of a Novel Non-Coding RNA Involved in Sperm Maturation*. PLoS ONE, vol. 6, page e26053, 2011. (Cited on page 123.)

[Oh 2006] J. D. Oh, H. Kling-Bäckhed, M. Giannakis, J. Xu, R. S. Fulton, L. A. Fulton, H. S. Cordum, C. Wang, G. Elliott, J. Edwards, E. R. Mardis, L. G. Engstrand and J. I. Gordon. *The complete genome sequence of a chronic atrophic gastritis Helicobacter pylori strain: evolution during disease progression*. Proc Natl Acad Sci U S A, vol. 103, no. 26, pages 9999–10004, Jun 2006. (Cited on page 102.)

[O'Rawe 2013] J. O'Rawe, T. Jiang, G. Sun, Y. Wu, W. Wang, J. Hu, P. Bodily, L. Tian, H. Hakonarson, W. E. Johnson, Z. Wei, K. Wang and G. Lyon. *Low concordance of multiple variant-calling pipelines: practical implications for exome and genome sequencing*. Genome Medicine, vol. 5, no. 3, page 28, 2013. (Cited on pages 89 and 97.)

[Otto 2012] C. Otto, P. F. Stadler and S. Hoffmann. *Fast and sensitive mapping of bisulfite-treated sequencing data*. Bioinformatics, 2012. (Cited on page 122.)

[Otto 2014] C. Otto, P. F. Stadler and S. Hoffmann. *Lacking alignments? The next-generation sequencing mapper segemehl revisited*. Bioinformatics, 2014. (Cited on pages 47 and 67.)

[Pabinger 2013] S. Pabinger, A. Dander, M. Fischer, R. Snajder, M. Sperk, M. Efremova, B. Krabichler, M. R. Speicher, J. Zschocke and Z. Trajanoski. *A survey of tools for variant analysis of next-generation genome sequencing data*. Briefings in Bioinformatics, page bbs086, 2013. (Cited on page 90.)

[Pandey 2008] V. Pandey, R. C. Nutter and E. Prediger. Applied biosystems solid$^{\text{TM}}$ system: Ligation-based sequencing, pages 29–42. Wiley-VCH, 2008. (Cited on page 3.)

[Phizicky 2010] E. M. Phizicky and A. K. Hopper. *tRNA biology charges to the front*. Genes Dev, vol. 24, no. 17, pages 1832–60, Sep 2010. (Cited on page 115.)

[Prüfer 2008] K. Prüfer, U. Stenzel, M. Dannemann, R. E. Green, M. Lachmann and J. Kelso. *`PatMaN`: rapid alignment of short sequences to large databases*. Bioinformatics, vol. 24, pages 1530–1531, 2008. (Cited on page 64.)

[Pushkarev 2009] D. Pushkarev, N. F. Neff and S. R. Quake. *Single-molecule sequencing of an individual human genome*. Nat Biotechnol, vol. 27, no. 9, pages 847–50, Sep 2009. (Cited on pages 2 and 5.)

[Quinlan 2008] A. R. Quinlan, D. A. Steward, M. P. Strömberg and G. T. Marth. *Pyrobayes: an improved base caller for SNP discovery in pyrosequences*. Nature Methods, vol. 5, pages 454–457, 2008. (Cited on page 8.)

[Ramakrishnan 2002] V. Ramakrishnan. *Ribosome Structure and the Mechanism of Translation*. Cell, vol. 108, no. 4, pages 557–572, 02 2002. (Cited on page 109.)

[Renda 2001] M. J. Renda, J. D. Rosenblatt, E. Klimatcheva, L. M. Demeter, R. A. Bambara and V. Planelles. *Mutation of the methylated tRNA(Lys)(3) residue A58 disrupts reverse transcription and inhibits replication of human immunodeficiency virus type 1*. J Virol, vol. 75, no. 20, pages 9671–8, Oct 2001. (Cited on page 114.)

[Richter 2012] J. Richter, M. Schlesner, S. Hoffmann, M. Kreuz, E. Leich, B. Burkhardt, M. Rosolowski, O. Ammerpohl, R. Wagener, S. H. Bernhart, D. Lenze, M. Szczepanowski, M. Paulsen, S. Lipinski, R. B. Russell, S. Adam-Klages, G. Apic, A. Claviez, D. Hasenclever, V. Hovestadt, N. Hornig, J. O. Korbel, D. Kube, D. Langenberger, C. Lawerenz, J. Lisfeld, K. Meyer, S. Picelli, J. Pischimarov, B. Radlwimmer, T. Rausch, M. Rohde, M. Schilhabel, R. Scholtysik, R. Spang, H. Trautmann, T. Zenz, A. Borkhardt, H. G. Drexler, P. Möller, R. A. MacLeod, C. Pott, S. Schreiber, L. Trümper, M. Loeffler, P. F. Stadler, P. Lichter, R. Eils, R. Küppers, M. Hummel, W. Klapper, P. Rosenstiel, A. Rosenwald, B. Brors and R. Siebert. *Recurrent mutation of the ID3 gene in Burkitt Lymphoma identified by integrated genome, exome and transcriptome sequencing*. Nature Genetics, vol. Vol. 44 (12), pages 1316–1320, 2012. (Cited on page 117.)

[Roberts 2013] N. D. Roberts, R. D. Kortschak, W. T. Parker, A. W. Schreiber, S. Branford, H. S. Scott, G. Glonek and D. L. Adelson. *A comparative analysis of algorithms for somatic SNV detection in cancer*. Bioinformatics, vol. 29, no. 18, pages 2223–2230, 2013. (Cited on page 13.)

[Roovers 2004] M. Roovers, J. Wouters, J. M. Bujnicki, C. Tricot, V. Stalon, H. Grosjean and L. Droogmans. *A primordial RNA modification en-*

zyme: the case of tRNA (m1A) methyltransferase. Nucleic Acids Res, vol. 32, no. 2, pages 465–76, 2004. (Cited on page 114.)

[Ross 1995] L. H. Ross, J. H. Freedman and C. S. Rubin. *Structure and Expression of Novel Spliced Leader RNA Genes in Caenorhabditis elegans*. Journal of Biological Chemistry, vol. 270, no. 37, pages 22066–22075, 1995. (Cited on page 83.)

[Rowold 2000] D. J. Rowold and R. J. Herrera. *Alu elements and the human genome*. Genetica, vol. 108, no. 1, pages 57–72, 2000. (Cited on page 119.)

[Salzman 2012] J. Salzman, C. Gawad, P. L. Wang, N. Lacayo and P. O. Brown. *Circular RNAs Are the Predominant Transcript Isoform from Hundreds of Human Genes in Diverse Cell Types*. PLoS ONE, vol. 7, page e30733, 2012. (Cited on pages 70 and 123.)

[Salzman 2013] J. Salzman, R. E. Chen, M. N. Olsen, P. L. Wang and P. O. Brown. *Cell-Type Specific Features of Circular RNA Expression*. PLoS Genet, vol. 9, no. 9, pages e1003777+, September 2013. (Cited on page 123.)

[Schmidtke 2012] C. Schmidtke, S. Findeiß, C. M. Sharma, J. Kuhfuß, S. Hoffmann, J. Vogel, P. F. Stadler and U. Bonas. *Genome-wide transcriptome analysis of the plant pathogen Xanthomonas identifies sRNAs with putative virulence functions*. Nucleic acids research, vol. 40, no. 5, pages 2020–2031, 2012. (Cited on page 125.)

[Schürmann 2007] K.-B. Schürmann and J. Stoye. *An incomplex algorithm for fast suffix array construction*. Softw., Pract. Exper., vol. 37, no. 3, pages 309–329, 2007. (Cited on page 35.)

[Shang 2014] J. Shang, F. Zhu, W. Vongsangnak, Y. Tang, W. Zhang and B. Sheni. *Evaluation and Comparison of Multiple Aligners for Next-Generation Sequencing Data Analysis*. BioMed Research International, vol. 2014, page 16, 2014. (Cited on page 122.)

[Shannon 1951] C. E. Shannon. *Prediction and entropy of printed english.* Bell Systems Technical Journal, pages 50–64, 1951. (Cited on page 60.)

[Sharma 2010] C. M. Sharma, S. Hoffmann, F. Darfeuille, J. Reignier, S. Findeiss, A. Sittka, S. Chabas, K. Reiche, J. Hackermüller, R. Reinhardt, P. F. Stadler and J. Vogel. *The primary transcriptome of the major human pathogen Helicobacter pylori*. Nature, vol. 464, no. 7286, pages 250–5, Mar 2010. (Cited on pages 101, 107, 109 and 125.)

[Smith 1981] T. Smith and M. Waterman. *Identification of common molecular subsequences*. Journal of Molecular Biology, vol. 147, no. 1, pages 195 – 197, 1981. (Cited on page 20.)

[Smith 2008] A. D. Smith, Z. Xuan and M. Q. Zhang. *Using quality scores and longer reads improves accuracy of Solexa read mapping*. BMC bioinformatics, vol. 9, page 128, 2008. (Cited on page 10.)

[Somel 2010] M. Somel, S. Guo, N. Fu, Z. Yan, H. Y. Hu, Y. Xu, Y. Yuan, Z. Ning, Y. Hu, C. Menzel, H. Hu, M. Lachmann, R. Zeng, W. Chen and P. Khaitovich. *MicroRNA, mRNA, and protein expression link development and aging in human and macaque brain*. Genome Res, vol. 20, no. 9, pages 1207–18, Sep 2010. (Cited on page 111.)

[Stricklin 2005] S. Stricklin*et al.* Wormbook, chapter C. elegans noncoding RNA genes. In The C. elegans Research Community [The C. elegans Research Community 2005], 2005. (Cited on page 83.)

[Swerdlow 2008] S. Swerdlow, E. Campo, N. Harris, E. Jaffe, S. Pileri, H. Stein, J. Thiele and J. Vardiman. World health organization classification of tumours of haematopoietic and lymphoid tissues, volume 2. IARC Press, Lyon, 4 édition, 2008. (Cited on page 117.)

[The C. elegans Research Community 2005] The C. elegans Research Community, editor. Wormbook. The C. elegans Research Community, http://www.wormbook.org, 2005. (Cited on pages 131 and 149.)

[Tomb 1997] J. F. Tomb, O. White, A. R. Kerlavage, R. A. Clayton, G. G. Sutton, R. D. Fleischmann, K. A. Ketchum, H. P. Klenk, S. Gill, B. A. Dougherty, K. Nelson, J. Quackenbush, L. Zhou, E. F. Kirkness, S. Peterson, B. Loftus, D. Richardson, R. Dodson, H. G. Khalak, A. Glodek, K. McKenney, L. M. Fitzegerald, N. Lee, M. D. Adams, E. K. Hickey, D. E. Berg, J. D. Gocayne, T. R. Utterback, J. D. Peterson, J. M. Kelley, M. D. Cotton, J. M. Weidman, C. Fujii, C. Bowman, L. Watthey, E. Wallin, W. S. Hayes, M. Borodovsky, P. D. Karp, H. O. Smith, C. M. Fraser and J. C. Venter. *The complete genome sequence of the gastric pathogen Helicobacter pylori*. Nature, vol. 388, no. 6642, pages 539–47, Aug 1997. (Cited on page 102.)

[Trapnell 2009] C. Trapnell, L. Pachter and S. L. Salzberg. *TopHat: discovering splice junctions with RNA-Seq*. Bioinformatics, vol. 25, no. 9, pages 1105–11, May 2009. (Cited on pages 70 and 127.)

[Trotochaud 2005] A. E. Trotochaud and K. M. Wassarman. *A highly conserved 6S RNA structure is required for regulation of transcription*. Nat

Struct Mol Biol, vol. 12, no. 4, pages 313–9, Apr 2005. (Cited on page 107.)

[Ukkonen 1985] E. Ukkonen. *Finding Approximate Patterns in Strings*. J. Algorithms, vol. 6, no. 1, pages 132–137, 1985. (Cited on page 22.)

[Ukkonen 1995] E. Ukkonen. *On-Line Construction of Suffix Trees*. Algorithmica, vol. 14, no. 3, pages 249–260, 1995. (Cited on page 32.)

[Van der Auwera 2013] G. A. Van der Auwera, M. O. Carneiro, C. Hartl, R. Poplin, G. del Angel, A. Levy-Moonshine, T. Jordan, K. Shakir, D. Roazen, J. Thibault, E. Banks, K. V. Garimella, D. Altshuler, S. Gabriel and M. A. DePristo. *From FastQ Data to High-Confidence Variant Calls: The Genome Analysis Toolkit Best Practices Pipeline*. Current Protocols in Bioinformatics, 2013. (Cited on page 12.)

[Vanet 2000] A. Vanet, L. Marsan, A. Labigne and M. F. Sagot. *Inferring regulatory elements from a whole genome. An analysis of Helicobacter pylori sigma(80) family of promoter signals*. J Mol Biol, vol. 297, no. 2, pages 335–53, Mar 2000. (Cited on page 109.)

[Vörtler 2010] S. Vörtler and M. Mörl. *tRNA-nucleotidyltransferases: highly unusual RNA polymerases with vital functions*. FEBS Lett, vol. 584, no. 2, pages 297–302, Jan 2010. (Cited on page 116.)

[Wang 2010] K. Wang, D. Singh, Z. Zeng, S. J. Coleman, Y. Huang, G. L. Savich, X. He, P. Mieczkowski, S. A. Grimm, C. M. Perou, J. N. MacLeod, D. Y. Chiang, J. F. Prins and J. Liu. *MapSplice: accurate mapping of RNA-seq reads for splice junction discovery*. Nucleic Acids Res, vol. 38, no. 18, page e178, Oct 2010. (Cited on pages 70 and 127.)

[Wang 2011] L. Wang, M. S. Lawrence, Y. Wan, P. Stojanov, C. Sougnez, K. Stevenson, L. Werner, A. Sivachenko, D. S. DeLuca, L. Zhang, W. Zhang, A. R. Vartanov, S. M. Fernandes, N. R. Goldstein, E. G. Folco, K. Cibulskis, B. Tesar, Q. L. Sievers, E. Shefler, S. Gabriel, N. Hacohen, R. Reed, M. Meyerson, T. R. Golub, E. S. Lander, D. Neuberg, J. R. Brown, G. Getz and C. J. Wu. *SF3B1 and other novel cancer genes in chronic lymphocytic leukemia*. N Engl J Med, vol. 365, no. 26, pages 2497–506, Dec 2011. (Cited on page 118.)

[Wang 2013] Q. Wang, P. Jia, F. Li, H. Chen, H. Ji, D. Hucks, K. B. Dahlman, W. Pao and Z. Zhao. *Detecting somatic point mutations in cancer genome sequencing data: a comparison of mutation callers*. Genome Med, vol. 5, page 91, 2013. (Cited on page 13.)

[Warren 2007] R. L. Warren, G. G. Sutton, S. J. M. Jones and R. A. Holt. *Assembling millions of short DNA sequences using SSAKE*. Bioinformatics, vol. 23, no. 4, pages 500–1, Feb 2007. (Cited on page 12.)

[Wassarman 2000] K. M. Wassarman and G. Storz. *6S RNA regulates E. coli RNA polymerase activity*. Cell, vol. 101, no. 6, pages 613–23, Jun 2000. (Cited on page 107.)

[Weese 2012] D. Weese, M. Holtgrewe and K. Reinert. *RazerS 3: faster, fully sensitive read mapping*. Bioinformatics, vol. 28, pages 2592–9, Oct 2012. (Cited on page 67.)

[Wen 2003] Y. Wen, E. A. Marcus, U. Matrubutham, M. A. Gleeson, D. R. Scott and G. Sachs. *Acid-adaptive genes of Helicobacter pylori*. Infect Immun, vol. 71, no. 10, pages 5921–39, Oct 2003. (Cited on page 107.)

[Wilda 2004] M. Wilda, J. Bruch, L. Harder, D. Rawer, A. Reiter, A. Borkhardt and W. Woessmann. *Inactivation of the ARF-MDM-2-p53 pathway in sporadic Burkitt's lymphoma in children*. Leukemia, vol. 18, no. 3, pages 584–8, Mar 2004. (Cited on page 118.)

[Williams 2000] M. A. Williams, Y. Johzuka and R. M. Mulligan. *Addition of non-genomically encoded nucleotides to the 3'-terminus of maize mitochondrial mRNAs: truncated rps12 mRNAs frequently terminate with CCA*. Nucleic Acids Res, vol. 28, no. 22, pages 4444–51, Nov 2000. (Cited on page 116.)

[Wilusz 2008] J. E. Wilusz, S. M. Freier and D. L. Spector. *3' end processing of a long nuclear-retained noncoding RNA yields a tRNA-like cytoplasmic RNA*. Cell, vol. 135, no. 5, pages 919–32, Nov 2008. (Cited on page 116.)

[Wu 2010] T. Wu and S. Nacu. *Fast and SNP-tolerant detection of complex variants and splicing in short reads*. Bioinformatics, vol. 26, pages 873–81, Apr 2010. (Cited on pages 71 and 127.)

[Xiong 2004] Y. Xiong and T. A. Steitz. *Mechanism of transfer RNA maturation by CCA-adding enzyme without using an oligonucleotide template*. Nature, vol. 430, no. 7000, pages 640–5, Aug 2004. (Cited on page 116.)

[Xu 2014] H. Xu, J. DiCarlo, R. Satya, Q. Peng and Y. Wang. *Comparison of somatic mutation calling methods in amplicon and whole exome sequence data*. BMC Genomics, vol. 15, no. 1, page 244, 2014. (Cited on page 89.)

[Yada 2001] T. Yada, Y. Totoki, T. Takagi and K. Nakai. *A novel bacterial gene-finding system with improved accuracy in locating start codons*. DNA Res, vol. 8, no. 3, pages 97–106, Jun 2001. (Cited on page 109.)

[Yap 2010] K. L. Yap, S. Li, A. M. Muñoz-Cabello, S. Raguz, L. Zeng, S. Mujtaba, J. Gil, M. J. Walsh and M.-M. Zhou. *Molecular interplay of the noncoding RNA ANRIL and methylated histone H3 lysine 27 by polycomb CBX7 in transcriptional silencing of INK4a*. Mol Cell, vol. 38, no. 5, pages 662–74, Jun 2010. (Cited on page 119.)

[Yu 2013] X. Yu and S. Sun. *Comparing a few SNP calling algorithms using low-coverage sequencing data*. BMC Bioinformatics, vol. 14, no. 1, page 274, 2013. (Cited on page 89.)

[Zerbino 2008] D. R. Zerbino and E. Birney. *Velvet: algorithms for de novo short read assembly using de Bruijn graphs.* Genome research, vol. 18, pages 821–9, 2008. (Cited on page 12.)

[Zhang 2013] Y. Zhang, X.-O. O. Zhang, T. Chen, J.-F. F. Xiang, Q.-F. F. Yin, Y.-H. H. Xing, S. Zhu, L. Yang and L.-L. L. Chen. *Circular Intronic Long Noncoding RNAs.* Molecular cell, vol. 51, no. 6, pages 792–806, September 2013. (Cited on page 123.)

Selbständigkeitserklärung

Hiermit erkläre ich, die vorliegende Dissertation selbständig und ohne unzulässige fremde Hilfe angefertigt zu haben. Ich habe keine anderen als die angeführten Quellen und Hilfsmittel benutzt und sämtliche Textstellen, die wörtlich oder sinngemäßaus veröffentlichten oder unveröffentlichten Schriften entnommen wurden, und alle Angaben, die auf mündlichen Auskünften beruhen, als solche kenntlich gemacht. Ebenfalls sind alle von anderen Personen bereitgestellten Materialen oder erbrachten Dienstleistungen als solche gekennzeichnet.

Leipzig, den 11. Juni 2014
gez. Steve Hoffmann

Dr. med. Dipl. Bioinf.

Steve Hoffmann

Curriculum Vitae

PERSONAL DETAILS

Birth	May 18, 1978
Birthplace	Gifhorn

CURRENT AND PAST POSITIONS

Junior Research Group Leader Transcriptome Bioinformatics 2009-today

LIFE Research Center for Civilization Diseases, University Leipzig

The Junior Research group with a total of 8 members is involved in a number of national and international projects and collaborations to develop novel computational methods for analysing data of high throughput sequencing experiments.

Member of the Board of Directors 2011-today

LIFE Research Center for Civilization Diseases, University Leipzig

Elected representative of young researchers at the LIFE Research Center.

Research Assitant 2007-2009

Research Assistant at the Bioinformatics group (head Prof. Stadler), University of Leipzig

EDUCATION

Doctoral Degree in Medicine (Dr. med.) 2007

Universität Düsseldorf

Dissertation on the genetic analysis of a family with autosomal-dominant hereditary nystagmus. This thesis was supervised by Prof. Bernhard Hemmer (now Head of the Department of Neurology at the Technical University of Munich)

Diploma in Bioinformatics (Dipl. Bioinf.) 2007

Universität Hamburg

The diploma thesis on Fast comparisions of protein structures using suffix arrays was supervised by Prof. Stefan Kurtz and Prof. Andrew Torda

Degree in Medicine (Ärztliche Prüfung) 2004

Universität Göttingen

The degree was obtained along with the german licence to practise medicine

Studies in Bioinformatics 2004-07

Universität Hamburg

Continuation of the studies in computer science with a special focus on bioinformatics at the Center for Bioinformatics at the University of Hamburg.

Studies in Computer Science 2001-04

Universität Marburg

The studies in computer science parallel to the studies in medicine were completed with a First diploma (Vordiplom)

Studies in Medicine 1998-07

Universität Marburg, Universität Goettingen

REVIEW AND ADVISORY PANELS

Peer Reviewer for Bioinformatics
Peer Reviewer for BMC Bioinformatics
Peer Reviewer for Algorithms in Molecular Biology
Peer Reviewer for Genome Biology
Peer Reviewer for Heredity
Peer Reviewer for Nucleic Acids Research
Peer Reviewer for Nature Biotechnology
Reviewer of Grant Proposals to the Swiss National Science Foundation (SNF)
Reviewer of Grant Proposals to the Government of Hong Kong
Member of the ECCB12 program comittee
Member of the ECCB14 program comittee

ACADEMIC RESPONSIBILITIES

Member of the Board of Directors 2011-today

LIFE Research Center for Civilization Diseases, University Leipzig

Elected representative of young researchers at the LIFE Research Center.

LANGUAGES

Languages	German (mother tongue) English (fluent) Spanish (advanced) French (basic)