

Julius-Maximilians-Universität Würzburg



Reducing the complexity of OMICS data analysis

Dissertation zur Erlangung des
naturwissenschaftlichen Doktorgrades
der Julius-Maximilians-Universität Würzburg

Vorgelegt von
Beat Wolf

aus Fribourg, CH, 2017

Eingereicht am: 5 April 2017
bei der Fakultät für Mathematik und Informatik

1. Gutachter: Prof. Dr. Thomas Dandekar
2. Gutachter: Prof. Dr. Pierre Kuonen

Tag der mündlichen Prüfung: 31 August 2017

Summary

The field of genetics faces a lot of challenges and opportunities in both research and diagnostics due to the rise of next generation sequencing (NGS), a technology that allows to sequence DNA increasingly fast and cheap. NGS is not only used to analyze DNA, but also RNA, which is a very similar molecule also present in the cell, in both cases producing large amounts of data. The big amount of data raises both infrastructure and usability problems, as powerful computing infrastructures are required and there are many manual steps in the data analysis which are complicated to execute. Both of those problems limit the use of NGS in the clinic and research, by producing a bottleneck both computationally and in terms of manpower, as for many analyses geneticists lack the required computing skills. Over the course of this thesis we investigated how computer science can help to improve this situation to reduce the complexity of this type of analysis. We looked at how to make the analysis more accessible to increase the number of people that can perform OMICS data analysis (OMICS groups various genomics data-sources). To approach this problem, we developed a graphical NGS data analysis pipeline aimed at a diagnostics environment while still being useful in research in close collaboration with the Human Genetics Department at the University of Würzburg. The pipeline has been used in various research papers on covering subjects, including works with direct author participation in genomics, transcriptomics as well as epigenomics. To further validate the graphical pipeline, a user survey was carried out which confirmed that it lowers the complexity of OMICS data analysis.

We also studied how the data analysis can be improved in terms of computing infrastructure by improving the performance of certain analysis steps. We did this both in terms of speed improvements on a single computer (with notably variant calling being faster by up to 18 times), as well as with distributed computing to better use an existing infrastructure. The improvements were integrated into the previously described graphical pipeline, which itself also was focused on low resource usage.

As a major contribution and to help with future development of parallel and distributed applications, for the usage in genetics or otherwise, we also looked at how to make it easier to develop such applications. Based on the parallel object programming model (POP), we created a Java language extension called POP-Java, which allows for easy and transparent distribution of objects. Through this development, we brought the POP model to the cloud, Hadoop clusters and present a new collaborative distributed computing model called FriendComputing.

The advances made in the different domains of this thesis have been published in various works specified in this document.

Zusammenfassung

Das Gebiet der Genetik steht vor vielen Herausforderungen, sowohl in der Forschung als auch Diagnostik, aufgrund des "next generation sequencing" (NGS), eine Technologie die DNA immer schneller und billiger sequenziert. NGS wird nicht nur verwendet um DNA zu analysieren sondern auch RNA, ein der DNA sehr ähnliches Molekül, wobei in beiden Fällen große Datenmengen zu erzeugt werden. Durch die große Menge an Daten entstehen Infrastruktur und Benutzbarkeitsprobleme, da leistungsstarke Computerinfrastrukturen erforderlich sind, und es viele manuelle Schritte in der Datenanalyse gibt die kompliziert auszuführen sind. Diese beiden Probleme begrenzen die Verwendung von NGS in der Klinik und Forschung, da es einen Engpass sowohl im Bereich der Rechnerleistung als auch beim Personal gibt, da für viele Analysen Genetikern die erforderlichen Computerkenntnisse fehlen.

In dieser Arbeit haben wir untersucht wie die Informatik helfen kann diese Situation zu verbessern indem die Komplexität dieser Art von Analyse reduziert wird. Wir haben angeschaut, wie die Analyse zugänglicher gemacht werden kann um die Anzahl Personen zu erhöhen, die OMICS (OMICS gruppiert verschiedene Genetische Datenquellen) Datenanalysen durchführen können. In enger Zusammenarbeit mit dem Institut für Humangenetik der Universität Würzburg wurde eine graphische NGS Datenanalysen Pipeline erstellt um diese Frage zu erläutern. Die graphische Pipeline wurde für den Diagnostikbereich entwickelt ohne aber die Forschung aus dem Auge zu lassen. Darum warum die Pipeline in verschiedenen Forschungsgebieten verwendet, darunter mit direkter Autorenteilnahme Publikationen in der Genomik, Transkriptomik und Epigenomik, Die Pipeline wurde auch durch eine Benutzerumfrage validiert, welche bestätigt, dass unsere graphische Pipeline die Komplexität der OMICS Datenanalyse reduziert.

Wir haben auch untersucht wie die Leistung der Datenanalyse verbessert werden kann, damit die nötige Infrastruktur zugänglicher wird. Das wurde sowohl durch das optimieren der verfügbaren Methoden (wo z.B. die Variantenanalyse bis zu 18 mal schneller wurde) als auch mit verteiltem Rechnen angegangen, um eine bestehende Infrastruktur besser zu verwenden. Die Verbesserungen wurden in der zuvor beschriebenen graphischen Pipeline integriert, wobei generell die geringe Ressourcenverbrauch ein Fokus war.

Um die künftige Entwicklung von parallelen und verteilten Anwendung zu unterstützen, ob in der Genetik oder anderswo, haben wir geschaut, wie man es einfacher machen könnte solche Applikationen zu entwickeln.

Dies führte zu einem wichtigen informatischen Result, in dem wir, basierend auf dem Model von „parallel object programming“ (POP), eine Erweiterung der Java-Sprache namens POP-Java entwickelt haben, die eine einfache und transparente Verteilung von Objekten ermöglicht. Durch diese Entwicklung brachten wir das POP-Modell in die Cloud, Hadoop-Cluster und präsentieren ein neues Model für ein verteiltes kollaboratives rechnen, FriendComputing genannt.

Die verschiedenen veröffentlichten Teile dieser Dissertation werden speziell aufgelistet und diskutiert.

Acknowledgment

For this thesis to happen and finish I have to thank numerous people and institutions. First and foremost I would like to thank Prof. Pierre Kuonen for not only giving me the opportunity to make this dissertation, but encouraging me to do so and giving me the best environment possible. I would also like to thank Prof. Thomas Dandekar for supervising my thesis, giving me precious advice and guidance in the field of bioinformatics. A big thanks goes also to Dr. David Atlan, that gave me the opportunity to perform this thesis with a very practical oriented approach, making it possible for much of my work being used in real laboratories across Europe. Having my work being used on a daily basis in a diagnostics environment was a major motivational force throughout the thesis. I would also like to thank Prof. Clemens Müller Reible and Prof. Simone Rost of the Institute of Human Genetics in Würzburg, for following my thesis with so much interest, giving me advice and most importantly for their trust in my work, introducing it in their laboratory to be used for the regular data analysis.

I would like to thank the co-authors with which I had the opportunity to write various papers, through which I could learn a lot and get familiarized with many topics. Without them, much of my work would be theoretical with no practical implications.

Having me supported me throughout the thesis, I also want to thank especially my girlfriend Gaëlle Kolly. A special thanks also goes to my parents, which made it possible to follow a research career.

Last but not least I would also like to thank the University of Würzburg and the University of Applied Sciences and Arts Western Switzerland for accepting me for my PhD. I'm grateful for having had the opportunity to make my PhD through a collaboration of two Universities, one more focused on the academic side and the other on the practical side.

Contents

1. Introduction	1
1.1. Motivation and scope	1
1.2. Contributions	3
1.3. Thesis outline	4
I. Foundations	5
2. Genetics	6
2.1. Introduction	6
2.1.1. Genetic code	9
2.1.2. Next generation sequencing	12
2.2. Summary	16
3. OMICs data analysis	18
3.1. Genomics	18
3.1.1. State of the art	21
3.2. Transcriptomics	27
3.2.1. State of the art	29
3.3. Epigenomics	31
3.3.1. State of the art	33
3.4. File-formats	35
3.5. Summary	37
4. Diagnostics	39
4.1. Introduction	39
4.2. Genetic disorders	41
4.3. Software requirements	43
4.4. Summary	45
5. Parallel & distributed computing	46
5.1. Introduction	46
5.2. History	47
5.3. State of the art	50
5.3.1. CPU	50
5.3.2. GPGPU	51
5.3.3. Distributed computing	52
5.4. Summary	53

II. Methods	54
6. Graphical pipeline	55
6.1. Introduction	55
6.2. Prototype	56
6.3. Methods	57
6.4. User interface	58
6.5. Project management	59
6.6. Annotations	60
6.7. Data analysis	62
6.7.1. Quality control	62
6.7.2. Sequence alignment	62
6.7.3. Coverage analysis	64
6.7.4. Variant analysis	65
6.7.5. Variant comparator	68
6.7.6. Copy number variations	70
6.7.7. Distribution	71
6.8. Discussion	72
7. Data analysis	73
7.1. Sequence alignment	73
7.1.1. Introduction	73
7.1.2. State of the art	74
7.1.3. Methods	75
7.1.4. Results	80
7.1.5. Summary	85
7.2. Meta-Alignment	86
7.2.1. Introduction	86
7.2.2. Method	87
7.2.3. Results	89
7.2.4. Summary	92
7.3. Variant calling	94
7.3.1. Introduction	94
7.3.2. State of the art	95
7.3.3. Methods	95
7.3.4. Results	98
7.3.5. Summary	102
7.4. RNA-seq	103
7.4.1. Introduction	103
7.4.2. Methods	104
7.4.3. Summary	108
7.5. Epigenetics	109
7.5.1. Introduction	109
7.5.2. State of the art	109
7.5.3. Methods	110
7.5.4. Summary	110
7.6. Genome browser	112
7.6.1. Introduction	112

7.6.2.	State of the art	113
7.6.3.	Methods	116
7.6.4.	Results	117
7.6.5.	Summary	125
7.7.	Discussion	125
8.	POP-Java	127
8.1.	Introduction	127
8.2.	State of the art	128
8.2.1.	Parallel computing in Java	128
8.2.2.	Distributed computing	129
8.2.3.	Language extensions	130
8.3.	POP Model	131
8.4.	POP Java prototype	132
8.4.1.	Limitations	133
8.5.	Implementation	134
8.5.1.	Annotations	134
8.5.2.	Additional changes	136
8.6.	Usage examples	136
8.6.1.	Distributed matrix multiplications	137
8.6.2.	Distributed mandelbrot	139
8.7.	Extensions	141
8.7.1.	Cloud integration	141
8.7.2.	Hadoop cluster integration	142
8.7.3.	TrustedFriendComputing	143
8.8.	Usage	146
8.8.1.	Distributed sequence alignment	146
8.9.	Limitations	149
8.10.	Discussion	149
III.	Applications	151
9.	Graphical pipeline applications	152
9.1.	Author participation	152
9.1.1.	Deep intronic variants in the factor VIII gene	152
9.1.2.	Myofibrillar myopathies	154
9.1.3.	Transcriptomics	155
9.1.4.	Epigenetics	156
9.2.	Indirect participation	157
9.3.	User survey	159
9.3.1.	Introduction	159
9.3.2.	Results	160
9.3.3.	Summary	164
9.4.	Discussion	164
10.	Conclusion & Future works	165

11. Publications	168
11.1. Publications	168
11.1.1. Journal papers	168
11.1.2. Conference proceedings	168
11.1.3. Misc.	169
11.2. Posters	169
 Appendices	 194
A. Artistic data visualization	195
B. Meta-alignment	200
C. Custom file formats	201
D. Polling	205
E. Downloads	210
F. Declaration of Authorship	211

1. Introduction

1.1. Motivation and scope

The field of genetics received a huge boost with the development of next generation sequencing (NGS) techniques which allow sequencing DNA (deoxyribonucleic acid) at speeds never seen before. A distinct research field, called bioinformatics, is dedicated to support geneticists to cope with the analysis of this data. While the domain of bioinformatics has been a part of biology and genetics in particular for a long time, it became an integral and indispensable part once the new sequencing started to create increasingly big amounts of data. Bioinformatics combines multiple fields, such as computer science, mathematics and statistics to solve the issues faced in biology. In the case of NGS data analysis, bioinformatics tools allow to automate a lot of analysis steps and to extract information from the data that would be impossible to do manually. This can happen in various domains using NGS data such as Genomics, Transcriptomics and Epigenomics, all grouped under the name of OMICS (alongside other -omics). While many tools exist to do various types of analysis for OMICS data analysis, their usage remains complicated and thus restricted to resourceful institutions. More often than not, the analysis of OMICS data requires the collaboration of bioinformaticians and geneticists, as both depend on the skill set of the other to analyze the data. This requirement for collaboration restricts the number of people that can work with NGS data and thus slows down the scientific progress of the field and keeps the costs high. Developing tools which allow geneticists to work independently from bioinformaticians becomes increasingly important as this dependency is a serious bottleneck in today's data analysis. While the acquiring of data continues to get cheaper (see Figure 1.1) and faster, the interpretation of the data did not yet follow at the same pace. This increase in time and cost to analyze the data comes not only from the complexity of the analysis itself, but also from the fact that the amount of data created increases faster than the computing capacities are predicted to improve. In 1965, Gordon Moore predicted the doubling of computing power in a single computer every 18 months. This is also known as Moore's Law and has been remarkably accurate (although arguably a self-fulfilling prophecy) ever since then. The evolution of Moore's Law and the reduction in sequencing cost is also shown in Figure 1.1, highlighting the problem of data analysis faced today.

To solve the computing power challenges associated with this evolution, a big focus in research is put on speeding up existing methods. When this is not possible, increasingly parallel and distributed computing is used to handle the large amounts of data produced. With the arrival of cloud computing this type of approach has been democratized by not limiting it to big organizations with access to grid environments or clusters. But the setup and usage of said tools is often complicated and requires a considerable amount of expertise in the domain. This again limits the number of people able to analyze genetic data and creates delays for the analysis.

The problem of accessibility is accentuated with genetic data analysis becoming increasingly common and affordable. Today, even private individuals can get their genetic data

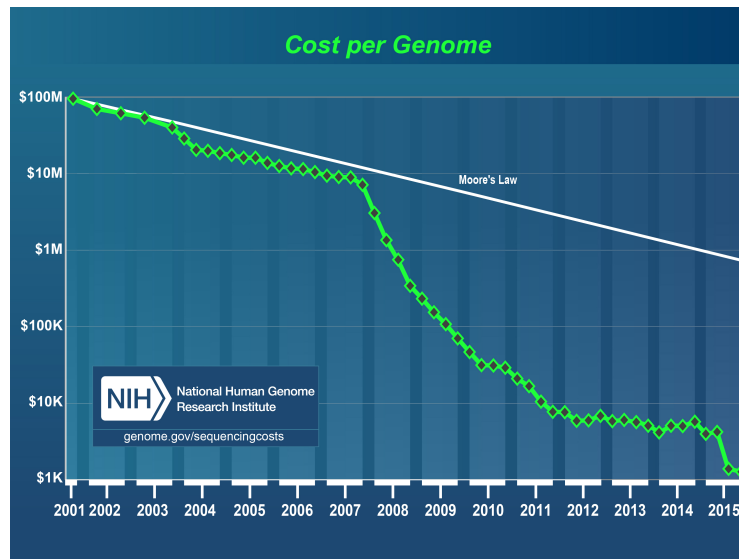


Fig. 1.1.: Graph of the radically dropping cost per genome, from <http://genome.gov/sequencingcosts>, 4 apr. 2017

through services like 23andMe¹ and sequencing technologies are being miniaturized, both to be cheaper and usable in remote locations [Hay15]. This leads to a possible future where many people have easy access to their own genome. This future also raises privacy issues with many of the available genetics analysis tools offloading the analysis to a cloud where the data security cannot be guaranteed. While private clouds, such as the one planned at the University of Würzburg, can mitigate those problems, they are not available to smaller laboratories or individuals.

This democratization and the possibility to analyze genetic data at home has various social implications. This thesis does not have as its goal to analyze this impact, but focuses on the field of clinical diagnostics, even if the improvements required in that field also enable easier research as well as genome analysis outside of laboratories.

The described problematics of the increasing data analysis complexity, the need for more accessible data analysis software and the computational infrastructure requirements, lead us to explore the following questions:

- How can NGS data analysis be made more accessible?
- Can the performance problems be solved by optimizing existing methods?
- Can distributed programming be made more accessible to facilitate distributed data analysis?

To approach those problems, we decided to develop a graphical NGS data analysis pipeline which aims at reducing the complexity of the data analysis. This graphical pipeline serves as an interface to the underlying tools, which can be either existing standard tools or re-implementations of existing methods in cases where for performance or other reasons it makes sense to re-implement them. We also use this graphical pipeline as the target for a distributed programming approach which aims at reducing the complexity of developing distributed applications.

¹<https://www.23andme.com>

We hope that the work being done in this thesis increases the number of people that are able to perform NGS data analysis and ultimately move the field forward. As with many fields, such as computer-science, the big advances were made once a larger public got access to the technology. Ideally, this work helps us to get one step close to this goal. The same is true for our work on POP-Java and the FriendComputing paradigm, which should enable for easier development of distributed applications and the usage of the cloud and other distributed environments. Be it for NGS data analysis or otherwise, we hope that the easier access to the technology will bring it to a larger community and thus ultimately move the field forward.

1.2. Contributions

During this thesis, multiple works have been published in journals and conferences. This section gives an overview of those contributions. A complete list of the published works in the context of this thesis can be found in Chapter 11, and every part of the thesis specifies the publications which have been made about the subject. We group the contributions according to the three research questions discussed in the previous section.

To explore how to make NGS data analysis more **accessible**, we published:

Beat Wolf, Pierre Kuonen, Thomas Dandekar, David Atlan, *DNaseq workflow in a diagnostic context, and an example of a user-friendly implementation*, BioMed Research International, Volume 2015 (2015), Article ID 403497, 11 pages

J. Elisa Bach, **Beat Wolf**, Johannes Oldenburg, Clemens R. Müller, Simone Rost, *Identification of deep intronic variants in 15 haemophilia A patients by Next Generation Sequencing of the whole factor VIII gene*, Thrombosis and Haemostasis, 2015: 114/4 (Oct) pp. 657-867

Meik Kunz, **Beat Wolf**, Harald Schulze, David Atlan, Thorsten Walles, Heike Walles and Thomas Dandekar, *Non-Coding RNAs in Lung Cancer: Contribution of Bioinformatics Analysis to the Development of Non-Invasive Diagnostic Tools*, Genes, Dec. 2016

Related to the **performance optimizations** of existing methods we published:

Beat Wolf, Pierre Kuonen, Thomas Dandekar, *GNATY: Optimized NGS variant calling and coverage analysis*, 4th International Work-Conference on Bioinformatics and Biomedical Engineering (IWBBIO 2016)

Beat Wolf, Pierre Kuonen, Thomas Dandekar, *Multilevel parallelism in sequence alignment using a streaming approach*, Second International Workshop on Sustainable Ultrascale Computing Systems (NESUS 2015)

Beat Wolf, Pierre Kuonen, *A novel approach for heuristic pairwise DNA sequence alignment*, BIOCOMP'13 - The 2013 International Conference on Bioinformatics & Computational Biology

And in regards to a more **accessible distributed programming** approach we published:

Beat Wolf, Pierre Kuonen, Thomas Dandekar, *POP-Java : Parallélisme et distribution orienté objet*, Compas2014, Conférence d'informatique en Parallélisme, Architecture et Système

Beat Wolf, Pierre Kuonen, Thomas Dandekar, *Comment reproduire les résultats de l'article: "POP-Java: Parallélisme et distribution orienté objet"*, Realis 2014: Reproductibilité expérimentale pour l'informatique en parallélisme, architecture et système

Beat Wolf, Monney Loïc, Pierre Kuonen *FriendComputing: Organic application centric distributed computing*, Second International Workshop on Sustainable Ultrascale Computing Systems (NESUS 2015)

This list does not include various other works, such as posters that have been published and can be found in Chapter 11.

1.3. Thesis outline

After the general introduction and overview given in this chapter, the rest of the document is organized into three main parts, Foundations, Methods and Applications.

In the first part, Foundations, the foundations of the thesis are laid down, giving the reader the required knowledge to understand the later chapters. As this document is written for readers with a computer science background, Chapter 2 explains the basics of genetics. This is followed by Chapter 3 about the different types of data analyses possible when analyzing NGS data. Those two general introduction chapters lead to Chapter 4, Diagnostics, which takes a look at how the data analysis is done in a diagnostic context. The foundations are closed with a Chapter 5 about parallel and distributed computing, a technology allowing us to analyze the big amount generated with NGS.

The second part, Methods, discusses the different applications and algorithms that were developed during this thesis. This part is split into three chapters, the graphical pipeline Chapter 6, the data analysis Chapter 7 as well as the POP-Java Chapter 8. While the first talks about graphical data analysis pipeline which was developed to make the analysis of NGS data more accessible, the second discusses specific NGS data analysis algorithms. The graphical data analysis pipeline integrates the various algorithms discussed in Chapter 7 in a comprehensive application targeting geneticists. In Chapter 8, the Java language extensions for easier distributed and parallel programming POP-Java is presented.

The third part of the thesis, Applications, discusses the practical use of the works presented in Methods. The focus is on the use of the graphical data analysis pipeline for various research projects and other contexts. Chapter 9 presents those works, as well as an evaluation of the impact of the graphical data analysis pipeline on both research and diagnostics work with NGS data.

Part I.
Foundations

2. Genetics

In this chapter we look at the basics of genetics required to understand later parts of this work. We look at how the genetic code is structured and how DNA works inside our cells. We then look at the current technologies that allow to bring the biological genetic code present in a cell into the digital world, which allows us then to perform data analysis. This technology, called next generation sequencing (NGS), is a central aspect of this thesis, as it is the source of most if not all data that we analyse. With this being an introductory chapter to genetics with no pretention to be complete, we encourage the reader to read specialized genetics books on the subject. We can highly recommend books like *Molecular Biology of the Cell* [AJL⁺14] which is updated constantly, *Molecular Biology* by Campbell, or *Essential Cell Biology* by Alberts B. et. al. Those books contain definitions and examples for all genetic terms, processes and methods discussed in this chapter.

2.1. Introduction

The study of genetics has a long history, but its modern form is a comparatively young science. Without the intention to discuss the full history of how genetics evolved over time, this chapter will start by giving a brief overview of how genetics started and where they are today. It will mainly focus on the aspects of human genetics, the evolution of our understanding of the human genome and sequencing technologies.

The word “genetics” comes from *genetikos*, an Ancient Greek word meaning “genitive” / “generative” which itself comes from the word *genesis* having the meaning “origin”¹. While an old word, it was William Bateson that mainly introduced the term genetics the way we use it today at the beginning of the 20th century (according to [Hay98]). He pushed the use of the term genetics to describe this newly evolving branch of science. Initially it was mainly used for the study of heredity and the mechanisms behind it, but over time also included the study of how the genetic material in cells actually works. But even if genetics only became a modern science in the early 20th century, parts of it are much older, such as the study of heredity, something that humans have been doing for a very long time.

It initially started with the selective breeding of animals and plants and later became a science in the modern meaning of the word. The first person to really approach the fields of genetics from a scientific point of view is Gregor Mendel, which many regard as the founder of modern genetics. While his initial work published in 1865 was not widely recognized in the scientific community, his work was rediscovered and recognized at the beginning of the 20th century. Over time, scientists discovered more and more about the inner working of cells and how they pass on their information to their offspring. DNA and the molecules which carry it, the chromosomes, have been discovered in the 19th century, but it was only in 1953 that James Watson and Francis Crick discovered the double helix structure of

¹<https://en.wikipedia.org/wiki/Genetics>

DNA. This finally revealed the nature of the molecule which holds the blueprint of all life. The DNA has a double helix structure and is a sequence of 4 different nucleotides adenine, cytosine, thymine and guanine also called A, C, T and G. A more detailed explanation of how the DNA is structured and how it works can be found in Section 2.1.1.

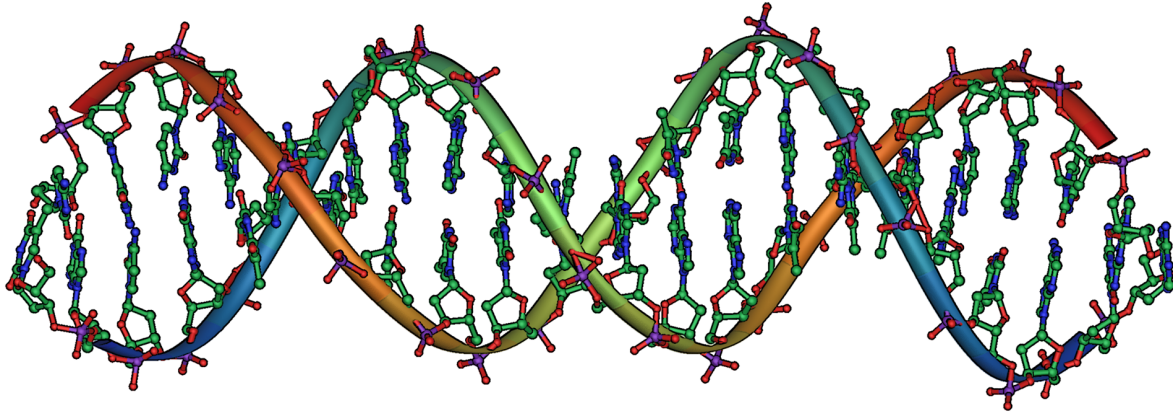


Fig. 2.1.: Visualization of the DNA double helix by Michael Ströck ²

While the structure and composition of the DNA were discovered at that time, its content was unknown and not yet readable. It was 1977 that Frederick Sanger invented chain-termination DNA sequencing, the first method to determine the actual sequence of nucleotides present in DNA and in turn the genes. This invention was key for the arrival of modern genetics as we know them today and which are also the focus of this thesis. The so called Sanger sequencing evolved from a research tool to a technology widely used in diagnostics. Other, similar methods to sequence DNA appeared at the same time, such as the Maxam-Gilbert sequencing.

Sequencing became increasingly better, but there was still one big obstacle when sequencing DNA and especially human DNA. It was unknown what the “average” human DNA looked like. This is a very important information to understand if a sequenced individual is “normal” or if it has a genetic difference that might explain its condition. While that information was known for small parts of the human genome, the sequence of the vast majority of the human genome was unknown. For this reason, the Human Genome Project[LLB⁺01] was started in 1990, with the goal to create a map of the human genome which would serve as a reference for further studies. The project officially achieved its goal in 2001 by providing the first human reference genome, a highly valuable resource for DNA sequence analysis. But even with the first official human genome having been released in 2001, the work on the project did not stop and continually releases new versions of the human genome. The new versions improve the completeness of the genome, as several big regions of the genome are currently not completely identified. Those regions often consist of highly repetitive regions which need a lot of effort to be put together. The last release of the human genome, HG38 (also called GRCH38) was released in December 2013. Even though the new human genome has been released for some time, it is not yet widely adopted, with most of the NGS data analysis tools still using HG19 (also called GRCH37).

²https://commons.wikimedia.org/wiki/File:DNA_Overview_landscape_orientation.png, Michael Ströck
CC-BY-SA-3.0

While the Human Genome Project worked to map the complete human genome, new sequencing technologies have been developed which greatly increased the speed at which the human genome could be sequenced. Those new technologies were regrouped under the term Next Generation Sequencing or in short NGS. We will look at those technologies in more detail in Section 2.1.2.

The arrival of next generation sequencing as well as the availability of a human reference genome lead to a new era of human genetics. While Sanger sequencing could be used to sequence complete genomes, as the Human Genome Project proved, this was a costly and time consuming process. In practice, Sanger sequencing was only used to sequence single exons or genes, but NGS greatly expanded those sequencing capabilities. NGS allows to sequence a genome at various levels, something that previous technologies could not do. The broadest level is called full genome sequencing, where all the regions of the genome are sequenced equally. This produces a very large amount of data, ranging into hundreds of gigabytes of data for a human sample. As most of the genome does not have any currently understood function, it is common practice to only sequence the parts of the genome which encode genes. This is called full exome sequencing, which amounts to over 10 times less data per sample as full genome sample. The lowest level is called targeted sequencing, where specific gene panels are sequenced. Those genes usually are specifically targeted because of their relevance to the data analysis question for the sample to be analysed.

Not only did NGS make all those different ways of sequencing possible, it also made it much faster than previous technologies. It became possible, in a matter of hours, to sequence complete gene panels, exomes or even full genomes. This development lead to many new discoveries, but also to new challenges. During the Sanger sequencing era, a lot of the data analysis could be done by hand. With the arrival of NGS this is no longer possible and requires new tools which can handle the big amounts of data produced by those new technologies.

While it is hard to determine what the future in sequencing will bring, there are still clear trends in the field. The trend of faster and cheaper sequencing technologies is set to continue. Additionally, there is also a trend in making the sequencing technology more accessible, making its use possible outside of genetic laboratories. Over the years, the commonly used sequencers became smaller and more efficient, with many now commonly used sequencers being barely larger than a desktop computer. Recently the improvements in that area took another big step with the MinION sequencer from Oxford Nanopore Technologies (See Section 2.1.2). This new kind of sequencer not only allows for impressing read lengths of 10'000 nucleotides or longer, but perhaps even more importantly, comes in the form of a device about the size of an external USB hard disk. While the quality of the technology may not yet be on the same level as commonly used sequencers [LHO⁺15], it already opens new opportunities to use the sequencing technology.

One such example is the analysis of the Ebola virus during the 2014 outbreak in Africa. By using the Oxford Nanopore MinION it was possible to sequence the Ebola virus directly on the field and to observe its evolution over time [Hay15]. This highly interesting use of DNA sequence technology is just one example of what might yet come. By miniaturizing the sequencing technologies as well as making them more accessible, one can easily imagine a future in which individuals can easily sequence their own genome at home. The issue of data analysis as well as data privacy will then become of even greater importance. And even without imagining a world in which people sequence their genome at home, the increased accessibility of the sequencing technologies already increases the potential users of this

technology. Smaller laboratories, private or public, increasingly use DNA sequencing to diagnose patients. While most of them are located in private institutions or hospitals, the mobile Ebola sequencing is a more extreme example of the new and upcoming possibilities of those new technologies.

As described by the Ebola sequencing efforts project, the MinION sequencing could indeed solve a big part of the hardware restrictions current approaches have. The smaller hardware used to sequence the Ebola virus, compared to a traditional sequencer, allows its usage directly on the field, without having to send samples to a laboratory. But the sequence analysis itself still remains a hurdle to overcome. Indeed, sequencing DNA is only part of the problem of creating a meaningful analysis of a sample. As mentioned previously, NGS produces a big amount of data per sample, with newer versions of sequencing technologies creating even more data. While the problem of sequencing DNA might be “solved” to some extent, or will be in the foreseeable future, this only moves the issue for genetic laboratories from sequencing to data analysis. Both in terms of the required computing infrastructure and the data analysis skills, new challenges need to be overcome to allow genetic laboratories, which might have done Sanger sequencing in the past, to analyze NGS data. Some of the data analysis infrastructure requirements can be outsourced using cloud computing or similar approaches. But this reliance on remote computing infrastructure not only causes problems when the internet connection is slow or unreliable, like in the Ebola example. Sending DNA data over the internet, especially from human samples, raises privacy concerns. As shown by Erlich and Narayanan [EN14], the identity of a person can at least be partially discovered, even with anonymized genetic data. Developing flexible and simple to use data analysis tools which can optimally use the existing infrastructure of a small laboratory as well as preserve the privacy of the patients is crucial for the adoption of NGS. Several of those approaches are discussed in Section 3.1.1.

The following Section 2.1.1 goes into more details on how DNA actually works inside the cell. This is then followed by Section 2.1.2 with a more detailed analysis of the current sequencing technologies which form the main source of biological information in this thesis.

2.1.1. Genetic code

This section gives a brief overview of how DNA works, from the current scientific point of view, especially in regard to human diagnostics. As Watson and Crick discovered, DNA is a molecule made out of four nucleotides, organized in a double stranded helix (see Figure 2.1). Both strands contain the same sequence but complementary. Complementary in the context of DNA means, that at if at a certain position in one strand the nucleotide A can be found, the other will always feature the nucleotide T. Same is true for C and G, making both strands copies of each other, but with the same information coded in different ways.

The DNA of a cell is organized in multiple chromosomes, in the case of humans those are 23 pairs of chromosomes. The number of chromosomes and the amount of copies varies from species to species, going from only 1 to several hundred chromosomes, with one to many copies of each. Humans have two copies of all chromosomes (except the Y chromosome). While one copy of those chromosomes comes from the mother and one from the father, they are not inherited exactly the way they are found in the parents. Instead, the chromosomes coming from both parents are combined through a process called chromosomal recombination, creating two distinct chromosomes with both a unique sequence. Both can contain unique genetic changes, an important fact for diagnostics which will be discussed in Section

4.2. Species with two copies of every chromosome, such as humans, are also called diploid and species with a single copy of each chromosome are called haploid.

Even if DNA is made out of only four basic building blocks, it is a highly complicated and poorly understood molecule. The one part of the genome which is currently best understood and studied is the functional part, called genes. Every chromosome contains a set of genes, which are special regions inside the chromosome. Those genes are responsible for the synthesis of RNA, a crucial molecule for the function of the cell. *RNA* (ribonucleic acid) is a molecule similar to DNA, so much that it can actually be described as a single stranded version of the DNA molecule. While ultimately this is not an accurate description, it does help to understand their relationship. RNA is, like DNA, built from four nucleotides: adenine, cytosine, uracil and guanine (A, C, U, G). We can see that those are in fact identical to the ones found in DNA, except for uracil (U) which in RNA is the equivalent of thymine (T) in DNA. Due to its chemical makeup, RNA is a less stable molecule than DNA. This means it has a much shorter lifespan, an important property for its role inside a cell.

RNA is a very flexible and versatile molecule. It is used for a variety of tasks, ranging from the creation of proteins to gene regulation (activating or deactivating individual genes). Compared to DNA, RNA is a very small molecule which can easily circulate inside the cell. This is why RNA and not DNA is used to make the cellular machinery work, the DNA contains the complete genetic information while the RNA is a copy of parts of DNA which does the actual work inside the cell.

There are different types of RNA, each with its specific role inside a cell. The most important type of RNA which we will encounter during this thesis, is the messenger RNA or also called mRNA. The mRNA is synthesized from a gene and is then translated into a protein. It contains the encoded protein sequence from which a protein will be built. Proteins play a key role in the cell, as pretty much everything inside a cell is made out of proteins, including the cell wall, the most visible part of a cell. Those proteins are what ultimately determine large parts of the *phenotype* of a person. The phenotype of the person is the sum of its observable aspects, be it physical or behavioral (including psychological). More about phenotypes and genetic disorders can be read in Chapter 4.

The process during which RNA is created from DNA is called *transcription*. During this process of transcription, parts of the region of DNA which contain the gene are used as a template to create a RNA molecule. The parts of DNA which are transcribed into RNA do not have to be uninterrupted. Certain parts of the gene are in fact not transcribed, the so called *introns*. The parts of the genes which are translated are called *exons*. Those exons contain coding sequences, which are translated to proteins, and untranslated regions, which are transcribed to RNA, but not the resulting protein. Those untranslated regions of the gene, also called *UTR*, have regulatory functions which control the frequency at which the gene is transcribed. Those UTR regions are located at the beginning and the end of the gene. During the transcription, a complementary copy of the DNA in the exons is created, creating the RNA. Figure 2.2 gives an overview of this process.

Most genes consist of more than one exon and can be transcribed in multiple ways. The different transcriptions will use a different set of exons (although always in the same order), to create different types of RNA. One possible combination of the exons is called a transcript, and every gene can have multiple transcripts, each having a different role in the cell. Most of the time a gene is expressed through one main transcript, but certain genes express more than one transcript or express specific transcripts in certain cell types (such as the liver, the brain etc.) Exactly how genes are activated and the individual transcripts are

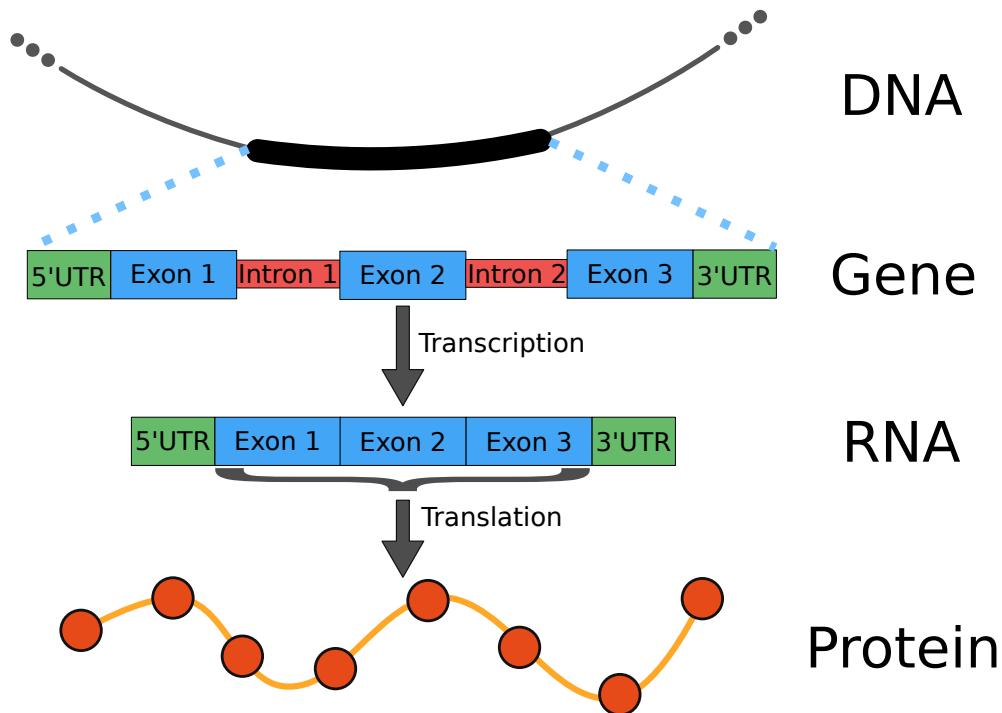


Fig. 2.2.: Going from DNA to proteins through the transcription of a gene to RNA which is then translated into a protein

selected is still an active topic of research. Without going into too much detail, genes have promoter sequences located near their start, in their untranslated region. Certain proteins interact with those promoter sequences to initiate the transcript event. An interesting new field of genetics, called epigenetics, studies the way the genes can be activated, or deactivated through a process called methylation. Chapter 3.3 discusses methylation in more detail.

This organization of DNA into chromosomes made up by genes, transcripts and exons can be seen as a way to organize data in a data-structure. With this thesis being done in computer science and thus undoubtedly being read by computer scientists, it can only be recommended to read the excellent document written by Bert Huber “DNA seen through the eyes of a coder”³. Some very intriguing comparisons are made between the way DNA and computer-code works.

One such example is how DNA stores the sequence of a protein to be transcribed. As discussed earlier, DNA is made out of 4 nucleotides, which can be compared to binary code, but using base 4 instead of base 2. Encoded using those 4 nucleotides is the information of how to construct a particular protein from a gene. As we have seen, the DNA is first translated to RNA, which in turn is used as the blueprint of the protein to be built. The way DNA encodes the protein is relatively intuitive from a computer science point of view. Just as ASCII code encodes one text character in binary as 8 bits (1 byte), DNA works with triplets of nucleotides called codons. One codon, which consists of 3 nucleotides, codes for one amino acid, the building block of a protein. A gene with a coding sequence of 300 nucleotides will for example create a protein with the of length 100 amino acids. An overview of all codons and what they code for can be seen in Table 2.1.

³<http://ds9a.nl/amazing-dna/>

TTT	Phenylalanine	TCT	Serine	TAT	Tyrosine	TGT	Cysteine
TTC		TCC		TAA		TGC	
TTA	Leucine	TCA		TAA	Ochre (Stop)	TGA	Opal (Stop)
TTG		TCC	TAG	Amber (Stop)	TGG	Tryptophan	
CTT		Proline	CCT	CAT	Histidine	CGT	Arginine
CTC			CCC	CAC		CGC	
CTA			CCA	CAA	Glutamine	CGA	
CTG			CCG	CAG		CGG	
ATT	Isoleucine	ACT	Threonine	AAT	Asparagine	AGT	Serine
ATC		ACC		AAC		AGC	
ATA	ACA	AAA		Lysine	AGA	Arginine	
ATG	Methionine	ACG	AAG		AGG		
GTT	Valine	GCT	Alanine	GAT	Aspartic acid	GGT	Glycine
GTC		GCC		GAC		GGC	
GTA		GCA		GAA	Glutamic acid	GGA	
GTG		GCG		GAG		GGG	

Tab. 2.1.: Codon table (DNA to Protein), taken from wikipedia ⁴

The codon table shows all possible combinations that can be made with 3 nucleotides. This amounts to 64 different codes that encode 25 different amino acids which form the protein. This shows an important aspect of the encoding scheme, which is the redundancy built into the encoding. A single change in coding DNA does not necessarily lead to a change in the protein coded by the gene. This allows the genetic code to have a certain degree of resistance to mutation, an event that can happen at every cell division.

Understanding how DNA works is key to understand biology, as it contains the blueprint of every cell. The best way to study it is of course to know the exact sequence of DNA which is in the cells of an organism. The following Section explores the technologies available today which are able to extract the DNA sequence from a cell and transform it into digital information that can be analysed.

2.1.2. Next generation sequencing

NGS is the big buzzword of recent years in genetics. When it came out in 2005, it allowed to move from the very limited Sanger sequencing, also called first generation sequencing, to a massive scale sequencing effort. This allowed to sequence complete genomes instead of just small parts of it, like with Sanger sequencing. Now the term NGS is not exactly new anymore, as the first references to it can be found as early as 2005 [Jar05]. In fact, there are multiple generations of sequencing technologies which have been released, and all sequencing technologies from the second one forward are regrouped under the NGS umbrella.

What all those newer generations of sequencing hold in common is their basic operating principle. Figure 2.3 gives an overview of this process.

The first step in NGS is the fragmentation of the DNA. During this step the DNA is cut into small pieces which are the source material which will be sequenced. The DNA is originally contained in chromosomes. But as chromosomes are too large to be read at once

⁴https://en.wikipedia.org/wiki/Genetic_code , status 6 sept. 2016

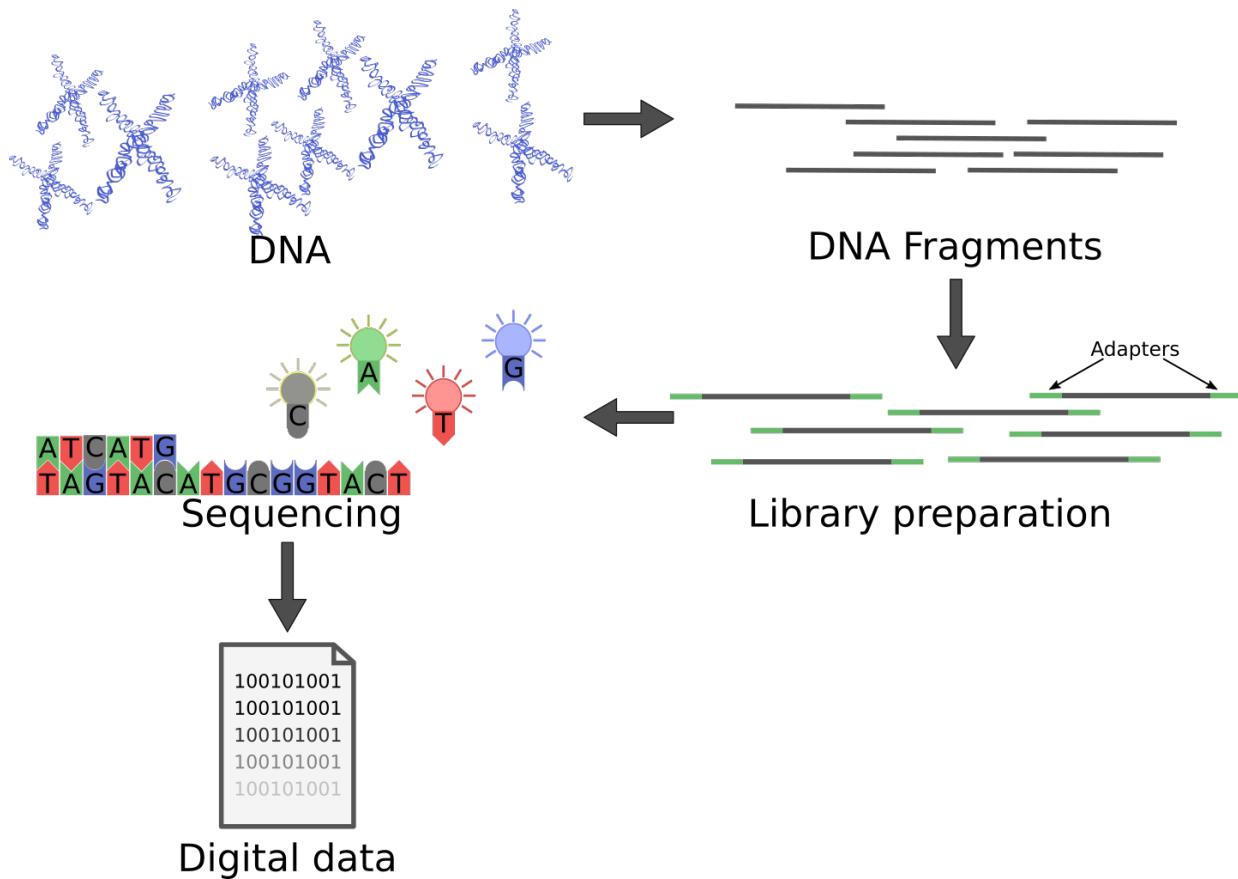


Fig. 2.3.: General NGS workflow

(they are between 50 and 250 million base pairs long for human chromosomes), they need to be split into smaller pieces which can be sequenced. There are different ways to split DNA into smaller pieces. For example through a physical process, like using ultrasound. Another way is to use specialized enzymes or chemical methods to split a large DNA molecule into smaller pieces. Before sequencing those fragments they go through a library preparation step, where the individual sequences are augmented by adapter sequences, also called tags. Those tags are used later for example to identify the sample from which the sequences comes when multiple samples are mixed during sequenced. After this step, all but the most recent NGS generations go through a process called *PCR* amplification (Polymerase Chain Reaction). This step allows to create clones of the individual sequences to be sequenced. After the PCR amplification step, the actual sequencing is performed.

Second generation sequencing, starting with the Illumina sequencers, sequenced reads of about 30 base pairs (bp). This value improved over time and current commonly used sequencing technologies sequence between 150 to 200 bp long reads, but latest developments push this limit up to 20'000 bp. The most important part to understand in NGS regarding sequence analysis is that the sequenced reads represent randomly chosen small parts of the original genome. This is also why this type of sequencing is sometimes called shotgun sequencing, as it recovers random sequences from all over the genome. While those regions can be restricted in targeted sequencing, the general principle of sequencing randomly selected regions of a genome stays the same even in that case. This collection of millions of randomly selected short reads from the genome are the main data source for current genetic analyses and form the basis of the data to be analyzed during this thesis.

As previously mentioned, NGS is not one single technology. The different sequencing technologies currently in use can be grouped into approximately two generations of NGS, generations two and three, as well as new and upcoming generations. We will also look at a third group, the future generations, which contains some experimental and exploratory technologies. Every sequencing method has its own particularities on how this process works, and we will briefly look into all of them in the following sections. For a more in-depth analysis of the different sequencing technologies it is highly recommended to read the excellent article written by Goodwin et. al. [GMM16].

Second generation

The first generation of next generation sequencers is also called the second generation of sequencers, with the first generation being the Sanger sequencers. The first sequencer machines in this generation came from Solexa/Illumina and Roche/454. 454 released their first sequencer in 2005, in collaboration with Roche. Roche acquired 454 2 years later in 2007 and continues to develop the 454 sequencing technology. Nearly in parallel, the first sequencer from Solexa, which was later acquired by Illumina, launched in 2006. Also in 2006, the ABI/SOLiD sequencing platform was launched, adding a third technology to the initial NGS generation. Figure 2.4 shows an example of a second generation sequencer, in this case an Illumina MiSeq sequencer.



Fig. 2.4.: Illumina MiSeq sequencer, image taken by Konrad Foerstner ⁵

All three technologies use similar, yet distinct approaches to sequence DNA. An excellent overview of the three technologies can be found in [Mar08]. They all sequence millions of sequences in parallel by putting them on so called flow cells. The sequences are analyzed by adding fluorescent chemicals that bind in the DNA nucleotides. After the addition of a specific chemical, a picture is taken to determine for every sequence what the current nucleotide is. The three sequencing platforms approach this problem slightly different, leading to platform specific error models. For example, the 454 sequencer has trouble determining the length of homopolymers (stretches of DNA in which the same nucleotide is repeated), as the light intensity measured at every cycle determines the amount of nucleotides of a particular type. SOLiD sequencing has a unique approach, where, at every step, 2 bases

are sequenced at once. The main problem with this approach is that it requires specially adapted analysis tools, as only very few analysis tools support this type of encoding.

The sequencers started with very low sequence lengths. Illumina and SOLiD started with read lengths around 30 bp long. Only the 454 sequencers had longer read lengths in their first generation, reaching on average 250 bp. Over time those sequencing technologies evolved leading to much longer sequences than where they were initially. The currently most used sequencing technology, which comes from Illumina, commonly sequences reads of 250 bp. Both SOLiD and 454 evolved as well, however they are much less relevant today. The improvements to the second generation of sequencers really allows to drive down the costs of sequencing. One example of this is the Illumina HiSeq X Ten platform, which promises a 1000\$ genome. Illumina themselves announced an even cheaper way to sequence, called Illumina NovaSeq, with a 100\$ target price for a genome.

Third generation

After the big success of the second generation of sequencers, the third generation started to be developed. Even with many improvements, the second generation had some major problems. One of them is the PCR step needed before sequencing the DNA. PCR not only adds a costly preprocessing step, but also introduces PCR related issues. One of the most important ones is the reduced capacity of sequencing regions of DNA with a high amount of G and C bases.



Fig. 2.5.: Oxford Nanopore Minion, image taken by Andrew Kilianski⁶

A common theme of the third generation sequencers like the Oxford Nanopore MinION (Figure 2.5) or the Pacific Biosciences (PacBio) SMRT (Single Molecule Real Time) is not to perform this PCR step. Instead, they directly sequence the original molecule found in the sample. While the DNA still has to be fragmented, any errors and limitations introduced by the PCR step can be avoided. They also both have in common that they provide much longer read lengths than the second generation of sequencers. But solving the problems of systematic errors introduced by PCR as well as the increased read lengths did not come

⁵https://en.wikipedia.org/wiki/File:Illumina_MiSeq_sequencer.jpg

⁶<http://blogs.biomedcentral.com/gigablog/2015/03/27/a-firsthand-perspective-of-trialling-new-mobile-dna-sequencing/>

without other problems. The error rates of both the PacBio and MinION sequencer are considerably higher than those of the previous generation. In the case of PacBio, those errors are random, which somewhat lessens the problem as they can be eliminated by increased sequencing depth, which increases the amount of times a certain position in the genome is sequenced. The error rates for PacBio are between 11% and 15% [RA15], with read lengths up to 60 kbp. The MinION sequencer is at the time of this not officially released, but has reported error rates around 15% [JFM⁺15] and read lengths over 100kbp for the longest reads.

In addition to the high error rates, the new generation of sequencer also does not yet achieve the throughput of the second generation, which means less DNA can be sequenced in the same amount of time. But the long reads allow to solve certain problems, like the analysis of repetitive regions in a genome, which is not possible with shorter read lengths. Another problem worth mentioning is the current lack of support for those new sequencing technologies by existing bio-informatics applications. This increases the complexity of the data analysis task compared to the second generation sequencers.

Future generations

In this chapter we take a brief look at what sequencing technologies might bring in the following years and how this will impact the way the data is analysed. Looking at the evolution of the sequencing technologies up to today and watching the announcements of the different sequencing technology manufacturers, some trends can be identified. One of the main trends is the simplification and cost reduction of sequencing. Even with the goal of a 1000\$ genome [Mar06] having now been more or less reached, sequencing will probably continue to become cheaper and less complex. This will not only allow more individuals to benefit from this new technology, but also increases the amounts of laboratories able to perform such analyses.

The other clear trend is more and improving data. As discussed earlier, sequencing length evolved from the initial very short reads to longer and longer reads. This trend to very long reads (over 100'000 bases) will likely lead to the ability to sequence entire chromosomes somewhere down the road. This would be a major change in the way the data analysis is approached. While today sequence alignment is a major part of OMICs data analysis, the increased read lengths will open up the possibility to use the more accurate method of sequence assembly. This would probably be a change of paradigm, requiring the complete replacement of most bio-informatics tools in place today. This will allow for a much more precise analysis of the genome of an individual. This includes a better understanding of structural variations, as well as phasing information. When a gene has multiple heterozygous variants it is currently very hard or even impossible to determine if the variants are on the same or different chromosomes. The only way to currently determine this information is through RNA sequencing as well as trio-analysis, where the child and both parents are sequenced. This information is very important to determine the actual protein encoded by a gene. With longer read lengths, this problem will be much easier to solve.

2.2. Summary

This chapter presented a general introduction into genetics, in particular in regards to how DNA works, how it is transcribed into proteins by going through RNA and how genetic

variations can alter this behavior. By presenting this process we also discussed how limited the current understanding of DNA is, a fact not surprising considering how young the science of modern genetics is and for how little time DNA can be sequenced. This lead us to the presentation of the various sequencing technologies available today, and most notably the NGS ones which are going to be our data-source going forward. The various sequencing technologies have different characteristics, like read lengths, error rates and throughput, making them more or less suitable for specific types of analyses. While most of them have a specific use-case, it has to be noted that most of the current sequencing is done using Illumina sequencers. Especially the latest iterations of the Illumina sequencing technology, like the HiSeq X and NovaSeq sequencers, which push the price point close to 1000\$ (or even 100\$) for one genome.

But there is also a different direction of research, equally interesting for the field, which is less focused on massive sequencing throughput but more on making it more accessible and deployable. This direction is mostly lead by the new MinION sequencers, which are able to sequence DNA in a much smaller form factor than existing sequencers, making their usage on the field or even at home possible. While the technological problems associated with this type of sequencing are not yet solved, it shows a possible future in which sequencing is much more accessible and common.

After having seen how DNA and RNA work and how they can be sequenced to be available in a digital form, the next chapter will look into more detail how that data can be analyzed.

3. OMICs data analysis

Once DNA has been transformed into digital data through the use of NGS technologies, it has to be analyzed. We look at three categories of data analysis which are possible with data generated through NGS.

The first is the sequencing and analysis of DNA with its four bases: A, C, T and G. This type of study is also called genomics and will be looked at in Section 3.1. Not only DNA can be sequenced with this type of technology, but also RNA, a molecule similar to DNA. Sequencing RNA is the basic data-source for the study of transcriptomics, the science of understanding what genes are transcribed in what quantity inside a cell. Transcriptomics are described in more detail in Section 3.2.

Last but not least, when studying epigenomics (Section 3.3), subtle changes in the DNA which are relevant to gene expression are analysed. All those different types of analyses can be grouped under the term OMICs, referring to their names which all end with *-omic*.

Other OMICs sciences exist but will not be covered in during this thesis. Examples of those field are lipidomics and proteomics. In lipidomics molecules called lipids are studied, for which an overview can be found in [AMZ⁺15]. Proteomics studies proteins and their role inside the cells. The interested reader is encouraged to read [BNK⁺14].

The following sections will look at the three different categories of OMICS data analysis. The types of analyses, how they are important and their current state of the art in terms of bio-informatics tools that support them are discussed. We also describe for all three OMICS data-sources a general workflow, which is put in a diagnostics context in Chapter 4.

3.1. Genomics

Genomics is the science that analyses the DNA of all living organisms. Today, the field of genetics in general is closely related to the sequencing technologies, which allow to study the actual DNA sequence found in any organism. Genomics includes the sequencing, assembly and study of genomes for different purposes. They can be scientific in nature, to further understand the structure and function of DNA, or medical, to perform risk assessment or diagnostic for genetic diseases (Chapter 4). Other uses, such as DNA profiling which is used to identify people, are also possible, but are not covered in this thesis.

Analysing genomics data can be done in various ways with different goals in mind. While recreating the sequence of the organism being studied is very often the main goal, the features of interest which are being looked at can determine how the analysis is done. One of the core disciplines of genomics is the reconstruction of an entire genome based on NGS data. The first complete genome of an organism was reconstructed in 1976, Bacteriophage MS2, through a tedious amount of manual work, way before any NGS technologies. While this was still feasible with a genome with the size of 3569 nucleotides, reconstructing more complex organisms like humans which have about 3 billion nucleotides requires more automated and effective approaches. The additional complexity not only comes from the length of the

genome, but also because humans have a diploid genome (2 copies of every chromosome) opposed to the haploid genome of Bacteriophage MS2.

To reconstruct a genome like the one of Bacteriophage MS2 or the human genome, nowadays a technique called *sequence assembly* is used. Sequence assembly combines the individual reads coming from NGS to reconstruct the original genome. Different techniques have been proposed over the years to approach the problem of sequence assembly. However, the techniques have a common underlying strategy to assemble a genome based on many millions of short reads with only a couple of hundreds bases. When sequencing a genome, many copies of the same chromosome are sequenced. Because of the randomness of the process, not all chromosomes are split up at the same locations when creating the sequence library which is sequenced. Because of this, the resulting reads will originate from different positions on the chromosome and overlap each other. The basic strategy to resolve this multi-million piece puzzle is to search for those overlaps and recreate the original genome by assembling the individual small reads to longer sequences. For a long time the *overlap–layout–consensus* approach was used to assemble sequencing data, like in GAP (Genome Assembly Program) [BSS95] to just name one representative of the early assemblers. The increasing amount of DNA sequences produced through NGS data and the downsides of the initial overlap–layout–consensus approach used for assembly, lead to the introduction of De Bruijn graphs. De Bruijn graphs were not a new concept when introduced by Pevzner et al. [PTW01] in 2001. In fact the De Bruijn graphs were initially discovered in 1946, and over half a century later their usefulness for sequence assembly was discovered. The De Bruijn graph based approach uses a special type of graph to solve the sequence assembly problem. The graph is constructed by splitting the sequenced reads into k-mers of a fixed size. A k-mer is a sequence of nucleotides of exactly length k. During this thesis, sequence assembly will not be used directly, but it is still important to understand how genomes are assembled. A more detailed overview about the different techniques and their limitations can be found in [HTN14]

What all those sequence assembly techniques hold in common is the high computational complexity of the task as well as the high amount of data that needs to be sequenced. Luckily, many other types of analyses do not require to assembly a complete genome from scratch, but their work is based on previous efforts. One goal of sequence assembly is the creation of a reference sequence for a species. One example is the Human Genome Project[LLB⁺01] which created the initial version of the human genome reference sequence. The reference sequence of a species does not actually represent the genome of one particular individual, but is a combination of multiple individuals. But as the difference between multiple individuals of the same species is relatively small (99.5% similarity between two humans [LSN⁺07]), such a reference sequence can be used to speed up the reconstruction of the DNA sequence considerably.

Instead of solving a problem in which millions of small sequences have to be compared between each other to find overlaps and reconstruct the original genome, every sequence has to be compared only with the reference sequence. This approach, which is called sequence alignment, or also mapping, consists in finding the position on the reference sequence which matches the sequence to align most. Searching for the best fitting position for a couple of hundreds nucleotides long sequence on a reference sequence of multiple billion nucleotides is not an easy operation, however it is still orders of magnitudes faster than performing a full assembly. During this process, the similarity of the sequence to align is determined with the reference sequence. The differences can be simple replacements of individual nucleotides

or deletions or insertions (indels) of parts of the sequence. The initial sequence aligner called BLAST which gained wide spread adoption was proposed in 1990 by Altschuhl et al. [AGM⁺90]. BLAST based its search on the assumption that any similarity of the search sequence and the reference sequence will contain small segments of perfect matches. Based on this assumption BLAST was able to quickly align sequences to large genomes, making the analysis of large quantities of DATA possible. BLAST received several updates over the years. While the initial version did not allow for gapped alignment (aligning sequences with insertions or deletions), later releases [AMS⁺97] addressed this issue. A multitude of sequence alignment tools were developed over the years each with its specialties and slightly different approaches. An excellent overview of those approaches can be found in [LH10]. With the development of many new tool (Section 3.1)s, BLAST (and its adaptations) remains one of the commonly used tool. But it is to note that around 2008, some new sequence aligners emerged that are now predominantly used. Most of those sequence aligners used a new technique based on a new type of index to quickly find potentially matching sequences between the query sequence and the reference. The technique for that new type of index was proposed in 1994 by Burrows and Wheeler [BW94] and was proposed to be used as a lossless data compression algorithm. As so often with algorithms adopted by the bioinformatics community, this new algorithm was not initially intended to be used with DNA, but generic data streams. The algorithm was called Burrows-Wheeler-Transform (BWT). This technique was used in 2008 by Lam et al. in an aligner called BWT-SW [LST⁺08], showing great results when used as the reference index. It was in 2009 that the aligners BWA [LD09] and Bowtie [LTPS09] appeared and pushed this technique further to accelerate sequence alignment. Both aligners are now widely in use, especially with their later improvements BWA-MEM [Li13] and Bowtie 2 [LS12a].

Sequence assembly and alignment are two of the main approaches to look at the data produced by NGS. Another approach is k-mer analysis, often used in metagenomics, a domain of research we are not presenting in this document. For the rest of the thesis, we will focus on sequence alignment as the method to analyze NGS data.

When doing NGS data analysis, typically a sequence of analysis steps extracts the relevant information from the raw sequencing data to produce the desired analysis result. In the context of diagnostics, the typical goal is to extract the variants present in a patient to create a final report which includes the probably pathogenic variants. Figure 3.1 shows an example of such a workflow as an UML activity diagram which goes from the raw sequencing data through a series of steps to produce a final variant report.

As depicted in Figure 3.1, the series of steps needed to extract the relevant information (in this particular use-case) is rather complex. Not only are there a series of analysis steps which depend on the output of the previous ones, but there are also often multiple alternative tools to perform the individual analysis steps. This process is not something the average user without special training in informatics can perform. The different tools required to perform the analysis steps are described in the next Section 3.1.1. A common approach to reduce the complexity and manual labor required to perform such an analysis is to use so called *pipelines*. NGS data analysis pipelines automate the process of data analysis, combining the different command line tools described in the following sections.

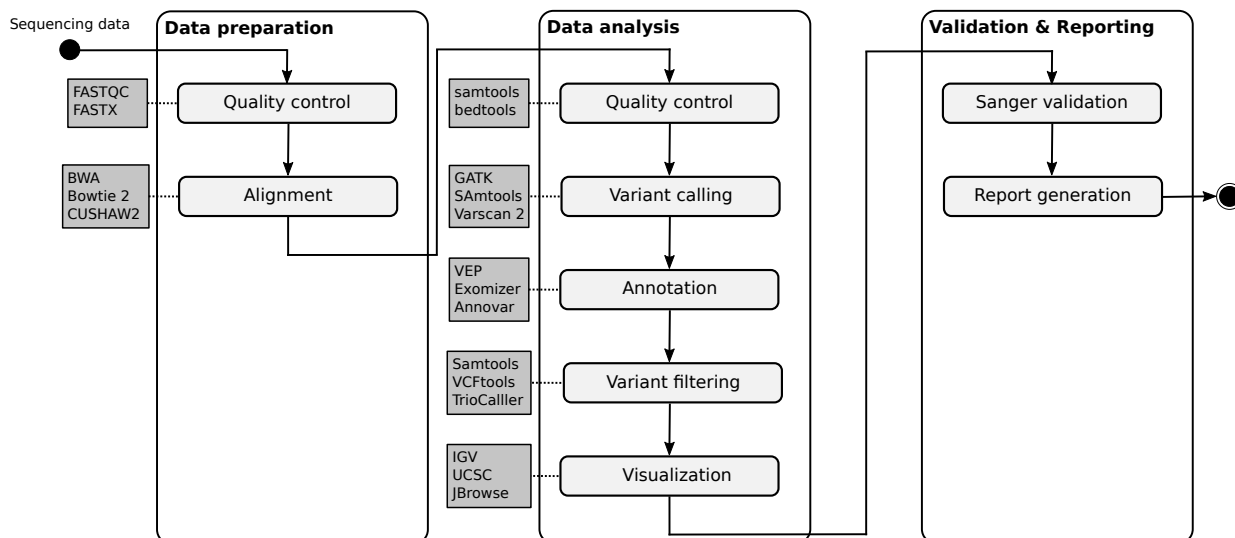


Fig. 3.1.: UML diagram of a NGS diagnostics workflow separated into different steps. A collection of the available tools for every step are indicated, based on the author paper [WKDA15a]

3.1.1. State of the art

The field of genomics produced a variety of tools and methods to analyze genetic data. Giving a complete overview of the tools used in the domain of genomics is an almost impossible task. There is a variety of tools with various specializations, with a constant stream of new methods being released. The following sections give an overview of the most important tools used in the field of genomics, in the context of this thesis. The structure of this section roughly follows the content of Figure 3.1, which describes the main workflow on which we base our work.

Raw data quality control

The first step of data analysis when doing NGS data analysis usually consists of a first step of quality control. Sequencing DNA is an error prone process where many things can go wrong. Making sure that the sequenced data corresponds to a certain quality standard is an important first step to not negatively affect downstream analysis. Several tools exist to perform quality checks of the raw sequencing data. This includes information like: Read count, read lengths, sequencing quality, duplicated read counts and more. A very commonly used application to perform this initial quality control is FASTQC¹. FASTQC can be used either as a command line tool or with a graphical user interface. Based on the quality control information gathered, the raw data is often filtered, for example to remove low quality reads or reads that are too short. This can be done for example with the FASTX toolkit [PWZM97], which not only allows to remove reads not fitting a minimum quality criteria, but also remove barcodes from reads. Barcodes are small pieces of DNA which are added to the DNA sequences before sequencing. Barcodes are also often called tags and are discussed more in Chapter 2.1.2.

¹<http://www.bioinformatics.babraham.ac.uk/projects/fastqc/>

Alignment

Sequence aligning is one of the key parts of NGS data analysis, at least in the context of this thesis. When aligning sequences against a reference sequence, millions of small sequences need to be compared with a one big reference. At the end of the process an alignment file is produced, usually in the BAM file-format (described in Section 3.4).

Different approaches exist to perform this task, this section provides an overview of the different tools and methods that exist. This section does not intend to be exhaustive, a more detailed analysis of existing aligners can be found in [LH10].

Sequence aligners require a reference sequence of the organism that is sequenced. Their job is to align the millions of the sequences that were sequenced against this reference. This essentially consists of finding the most probable place from which the sequence originates from, with the assumption that the reference sequence closely matches the actual sequence in the referenced sample. Finding the most probable place consists of finding the location on the reference at which the distance between the sequence and the reference is as small as possible. This distance is essentially the edit distance, counting the number of mismatches, insertions and deletions required to match the reference at a given position. Several algorithms exist to solve this problem with most notably Smith-Waterman [SW81], Needleman-Wunsch [NW70] and Gotoh [Got82] being used today. All of those algorithms are now relatively old (they were developed in the 1970-80) and are well understood. All three algorithms are dynamic programming algorithm, which divide the problem in smaller sub problems which can be solved faster by using the information of previously solved problems.

What those edit distance algorithms all have in common is that they are computationally expensive, too costly to be applied on the full reference genome for the millions of sequences to be aligned for every sample. This is why a series of heuristic is applied to reduce the search space. The usage of those heuristics is the main difference between the different aligners that exist today. To reduce the search space in the reference sequence, aligners use so called indexes. A reference index can be compared to an index in a book, which allows to quickly look up the positions of interest in the reference sequence. An example of such an index is a so called hash index, where all the positions of a k-mer of a certain distance are saved. For example, with a k-mer length of 3 to build the index, it would be possible to quickly find all positions in the reference at which the sequence ATC can be found. This information is then used to for example only test certain parts of the reference for a match with a specific sequence, using the previously described dynamic programming algorithms.

The most known aligner, BLAST [AGM⁺90], uses this method with a default k-mer size of 11 nucleotides.

Another more recent method for reference indexes is the use of the Burrows-Wheeler-Transformation (BWT), which was initially developed as an algorithm to be used in the context of data compression [BW94]. This method was adapted to sequence alignment to both efficiently store the reference sequence in memory, as well as to query it to find matching regions between the reference and the sequence to be aligned. Most notably the Burrows-Wheeler aligner [LD09] (BWA) uses this technique to accelerate sequence alignment. But other aligners, such as Bowtie 1 [LTPS09] and 2 [LS12a], as well as CUSHAW2 [LS12b] use this approach.

Sequence alignment, even with the use of many methods that help to accelerate it, is still the most time consuming process of NGS data analysis. This is why over time, different additional approaches have been developed to accelerate sequence alignment, notably by

using parallel and distributed computing architectures.

NVIDIA, the leading manufacturer of high end graphics cards (also called GPUs), developed the NVBIO² framework, making it easier for programmers to use the power of GPUs when developing bioinformatics applications. One example of this effort is the nvBowtie aligner, which is a reimplement of the popular Bowtie 2 aligner, but using the GPU through the usage of CUDA (Compute Unified Device Architecture) to accelerate the alignment. Not only nvBowtie integrates the usage of GPUs, but to give another example of the previously mentioned aligners, CUSHAW2 has a GPU counterpart called CUSHAW2-GPU [LS14].

To lower the time to align a particular sample, it is only natural to turn to parallel distributed computing. Several methods to distributed sequence aligners over multiple computers have been developed over time. While often the “manual” distribution method is used, where the raw data input is split equally over multiple computers and the resulting alignment file is merged at the end. But more advanced distribution methods have been developed, with notably Hadoop[Whi12] being a popular platform for the distribution. Both CloudBurst [Sch09a], which distributes a sequence aligner based on RMAP [SCH⁺09b] and BigBWA [APPA15], which distributes the BWA aligner, use Hadoop as their distribution platform. Other approaches exist, such as ScalaBlast [OB13], which distributes the Blast aligner using the Scala [Oa04] programming language. Setting up and maintaining an infrastructure which can run such a distributed aligner remains a complicated task, especially if cloud solutions need to be avoided for privacy reasons.

Last but not least, there is a special category of aligners, so called realigners, which intend to solve a particular problem in sequence alignment. Aligning sequences with indels is an error prone process. As every sequence is aligned independently against the reference sequence, this can lead to incoherent results around indels, depending on where the indel is located inside a particular sequence. Indel realigners try to solve this issue by making a second pass at alignment, after the initial alignment is done. During this process, all sequences around a potential indel are aligned against each other to provide a more coherent result. This makes it easier for downstream tools, like the variant callers discussed in the next Section 3.1.1 to detect those indels. GATK [MHB⁺10], which stands for Genome Analysis Toolkit, is a collection of tools, which among other tools, contains a tool to perform realignment.

As one can imagine, the aligner and settings used during the sequence alignment have an impact on the rest of the analysis, such as the variant calling discussed in Section 3.1.1.

Quality control

Before analyzing the aligned data, another round of quality control is common when doing NGS data analysis. It is only at this step, that it can be determined how well the sample was sequenced and if there were any contaminations or sequencing problems in particular regions of the genome. Two main tools allow to determine the quality of an alignment, samtools [LHW⁺09] and bedtools [QH10]. One of the easier metrics to obtain is the number of aligned reads, which is a first good measure of the alignment quality. Another important measure is the coverage, also called sequencing depth, of the different regions that were sequenced. This allows to determine if all regions of interest have been sequenced and aligned correctly.

The result of this quality control will determine if the sample needs to be resequenced

²<http://nvlabs.github.io/nvbio/>

or not, and if there are restrictions in terms of the data analysis, for example for badly sequenced genes.

Variant calling

Variant callers are the main tools after sequence aligners when analyzing NGS data in genomics. After a sequence aligner aligned millions of reads to the most probable location on the genome, the most interesting question to ask is where there are differences between the aligned data and the reference sequence. As the human genome is about 3 billion basepairs long, and there is an approximate 99.5% similarity between the DNA of two humans [LSN⁺07], this amounts to about 15 million differences which can be expected to be found. Searching the aligned genome by hand for those differences is of course not feasible, especially as there are a number of differences which come from sequencing artifacts as well as alignment errors. In this section we only consider variant calling for small changes in the genome, like SNPs (single-nucleotide polymorphism) or indels (insertions/deletions), but not for structural variations (SVs). The analysis of copy number variations (CNVs) which are one type of SVs, is discussed in Section 6.7.6.

To automate the process of finding true variations in a sample genome, variant callers are used. Various approaches exist to determine if the sequenced genome contains a variant at a specific position. The two main approaches are to either use heuristic/statistical approaches or probabilistic methods.

Two of the main tools used today are in the probabilistic category. GATK [DBP⁺11] and samtools [LHW⁺09] both use the Bayes theorem to determine if there is a variant at a certain position on the genome.

The most used variant caller in the heuristic/statistical category is Varscan 2 [KZL⁺12]. Through a user set list of filters, such as a minimum observation frequency or coverage, variants are called.

You can find a more detailed look at variant callers in Section 7.3 which also presents a variant caller developer during this thesis.

Variant analysis

Once variants have been called, they need to be annotated to put them into their biological context to be more easily interpreted. This makes it easier to find the variants of interest for a specific sample as well as their effect on it. Annotating variants augments every variant with the information relevant to the question asked about the dataset. On the most basic level, the information of what genes are affected by a specific variant is determined. This can be further expanded by adding information like the effect the variant has on the protein which is transcribed by that particular gene. An example of this would be the addition or removal of a stop codon as well as the change of an amino acid in the protein.

There are two other types of annotation which are commonly added during the annotation phase. The first one is the information about a variant in public databases. This includes information like how frequently a variant has been observed in the general population as well as documented consequences of those variants, for example in human diagnostics. The second type of information added is based on predictions of how a variant affects the transcribed protein, but on a level which can be less easily determined, as for example the addition of a stop codon. Those predictions are often based on information if there might be

a disturbance in the way the exons are transcribed or based on conservation scores, which indicate how preserved a certain amino acid is across various species.

Different tools exist to annotate variants with one or more of the listed information. Just to name a few, the Ensembl variant effect predictor (VEP) [MPR⁺10], which is a webservice which adds all sorts of annotations to variants, based on the information available in the ensembl database. A similar approach is used by exomizer [RKO⁺14], a Java application developed by the Sanger institute, which can be run locally and annotates and prioritizes variants on various annotations. This list would not be complete without annovar [WLH10], which features a similar feature set to exomizer.

After having annotated the variants, the most common thing to do is to filter them according to some criteria, for example their effects on the organism or their frequency in the general population. The variants are usually stored in VCF file (see Section 3.4), which can be filtered using the VCFTools [DAA⁺11] program. This application allows to filter and manipulate VCF files to extract the variants of interest.

For more advanced, like identifying de novo mutations in a child for which both parents have been sequenced (also called trio analysis), tools like TrioCaller [CLZ⁺13] are used.

Visualization

Visualizing genomics data is an integral part of the data analysis, both to better understand the data as well as to verify the results of automated data analysis. Various ways exist to visualize the data, with some being more or less practical (see Appendix A for an original artistic approach we developed). But the most common way to visualize genomic data is through so called genome browsers, which as their name suggest, are used to visualize and browse a genome. This includes most commonly the sequence alignment of one or multiple samples as well as the integration of additional information.

Various browsers exist nowadays, specializing in different domains. The two large groups of genome browsers are web based genome browsers and those that run as desktop applications. Both groups have different use cases and are able to display a variety of information.

Two of the most used web based genome browsers are the UCSC (University of Santa Cruz) genome browser [KSF⁺02] and the Ensembl genome browser [CAB⁺15]. Both of those tools are highly popular to browse a genome with a variety of additional genome annotations and customizable annotation tracks. Also interesting to mention are more specialized tools, like JBrowse [BYD⁺16], which provide a genome browser that can easily be integrated into an existing website.

In the other group, the Integrative Genomics Viewer (IGV) [TRM13] is one of the most used desktop applications. Another one is Tablet [MSB⁺13], which is more focused on sequence assemblies of NGS data.

More details about genome browsers can be found in Section 7.6 where the genome browser developed during this thesis is presented.

Analysis pipelines

A more streamlined approach to genetics data analysis can be achieved through the use of so called analysis pipelines. When doing NGS data analysis, most of the used tools are used on the command line, just like most of the tools described in the previous sections. Those tools are then combined, by using the output of one tool and using it with another, to perform the desired data analysis. This is a complicated process which requires a lot

of experience with command line tools in general, and the tools used for the analysis in particular. The often incompatible data-formats which need to be converted add to the complexity of the task. As a consequence, the amount of people that are able to directly work with the data is heavily reduced.

To render the process more approachable and efficient, automated and semi-automated analysis pipelines have been created. The main task of those tools is to abstract the underlying complexity of the analysis from the user. Through nice user interfaces or automation, the user does not have to manually perform the different analysis steps. We present here a short overview of some of the key pipelines used today, with their characteristics. We group the presented pipelines into three categories. Desktop based solutions which are locally installed on the computer running the analysis. Web based solutions which either run on the cloud or on a centralized server inside or outside the laboratory which use them. And we finish the list with frameworks which allow to create custom data analysis pipelines, usually without graphical user interfaces, an option still largely used by many laboratories.

The following pipelines listed are desktop based.

CLCBio : CLCBio ³ is a big and popular OMICs analysis toolsuite. It covers most aspects of modern OMICs analysis, including genomics. The main feature of CLCBio is the ability to not only use a predefined analysis pipeline, but the ability to create a customized pipelines based on custom designs. The design principle of CLCBio is based on the ability to “draw” an analysis pipeline to build a pipeline specifically for the experiment to be conducted. This is done by providing a multitude of independent analysis modules with standardized inputs and outputs. The user can then connect those modules, as well as configure them, to create an automated analysis that can be reused multiple times.

NextGENe : NextGENe ⁴ is another full-fledged OMICs analysis toolsuite. It is developed by Softgenetics and is locally installed on the user’s computer. NextGENe is part of a larger collection of specialized genetic data analysis software and uses custom algorithms for the different analysis steps.

The pipelines listed next are web based.

Cartagenia Bench Lab: Cartagenia, which is part of Agilent Technologies, developed a commercial software aimed at clinical genetics ⁵. The pipeline is mainly focused on the annotation and interpretation of variants found in a sample. Cartagenia Bench Lab is a cloud based software which integrates into existing workflows for clinical diagnostics. To address privacy issues, the software can either run on a public or a private cloud, so that no sensible data leaves the laboratory.

Galaxy : Galaxy [GRH05] is a web based analysis framework. In contrast to the previously described tools, Galaxy runs on a webserver. Users upload their data to the server and also analyze the data on that same server. Similarly to CLCBio, Galaxy is composed of many independent data analysis modules. Users can combine those modules to create an automated pipeline, which they can reuse at a later point. Galaxy is an open source project

³<http://www.clcbio.com/>

⁴<http://www.softgenetics.com/NextGENe.php>

⁵<https://cartagenia.com/cartagenia-bench-lab>

that can be used freely. It focuses mostly on the analysis part, and does not visualize the data for the users. To visualize the data, external tools like the UCSC genome browser [KSF⁺02] are integrated.

DNAnexus : DNAnexus ⁶ is another web based analysis framework. But in contrast to Galaxy, DNAnexus specializes on the cloud, offering laboratories to offload all the data analysis to a decentralized scalable cloud environment. Like Galaxy, the users upload their data to DNAnexus, and analyze it on a remote server.

Sophia-DDM : Sophia DDM ⁷ is an online application developed by Sophia Genetics. It provides a software as a service approach to do NGS data analysis in diagnostics, providing an integrated solution for hospitals.

Some other new web based NGS analysis tools are Geneious ⁸ and Opal Clinical ⁹, which are developed by startups for the NGS diagnostics market.

We complete the list with frameworks which allow to create custom pipelines, chaining together multiple data analysis steps to automate the process of NGS data analysis.

While any command line tools on a Unix like system can be automated and chained together using shell scripts, some more sophisticated approaches exist to create a typical NGS data analysis pipeline. One of the more popular, python based, tools to manage the workflow of a pipeline is Snakemake [KR12]. It allows the user with a simple syntax to define different data analysis steps and chain them together. Through its flexibility it allows experienced bio-informaticians to develop and maintain an automated NGS data analysis pipeline.

Another example of a python based framework to automate custom pipelines would be Ruffus [Goo10]. Ruffus was developed, like Snakemake, with the bio-informatics community in mind.

3.2. Transcriptomics

After Genomics, Transcriptomics is another popular field for both research and diagnostics that uses NGS technology. This section intends to give an overview of what transcriptomics is as well as what techniques are used, especially in regards to data analysis. The presented methods and use-cases are focused on what is relevant for the rest of this thesis. For a more in-depth look into the field, the excellent survey article by Conesa et. al. [CMT⁺16] as well as the books *RNA-seq Data analysis, A practical approach* by Korpelainen et. al. and *Transcriptomics, Expression pattern analysis* by Gomase, V et. al. are recommended to the reader.

The term “transcriptomics” comes from the word transcription, which is the process during which DNA is transcribed into RNA. As we have seen in Section 2.1.1, the DNA in every cell is used as the blue print for RNA. This RNA can have different purposes, with the most important one for this thesis being the creation of proteins. The transcribed RNA represents actual biological events inside the cell, as opposed to the DNA where the biological consequences of many modifications are unknown.

⁶<https://www.dnanexus.com/>

⁷<http://www.sophiagenetics.com/hospitals/sophia-ddm/sophia-ddmtm-details.html>

⁸<http://www.geneious.com/>

⁹<http://www.omicia.com/products/opal-clinical/>

RNA and DNA, when in the single stranded form, are very similar molecules. So similar in fact, that the NGS technologies can sequence both of them. The sequenced RNA consists of sequences from many thousands of different RNA sequences in a cell.

Analysing sequenced RNA data can have different purposes, with two main types of analyses being common.

The first is to analyze activity of the various transcripts in a sample, focusing on known transcripts. The determination of how much every transcript/gene is expressed in the sample to analyse, also called quantification. In this step, the individual NGS sequences are linked to the various known transcripts, which makes it possible to rank them in order of activity. In a second step, those values are often compared between samples, which is also called expression analysis, where the expression of genes, as determined by the quantification step, is compared between samples. This makes it possible to determine the gene activity across different cell types or scenarios, to analyze how the transcriptome inside a cell behaves depending on the situation.

Figure 3.2 shows the general workflow for this type of analysis. We can see that the basic steps are very similar to the genomics workflow, even with many of the tools being used in both approaches.

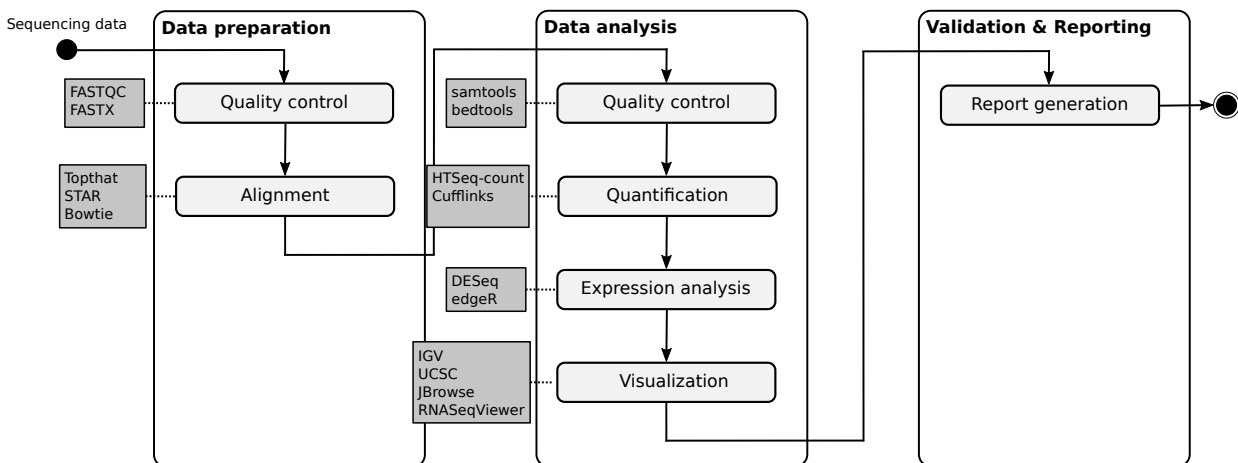


Fig. 3.2.: Transcriptomics workflow separated into different steps. A collection of the available tools for every step are mentioned, based on author paper [WKDA15a]

This workflow does of course not cover all aspects of RNA-seq data analysis, including the second type of analysis being done with this type of data. The second type of RNA-seq data analysis is the discovery of new transcripts or even genes. This is done by discovering RNA sequences which do not originate from any known gene or show alternative splicing of existing genes. This type of analysis is not going to be looked at in detail in this document.

It is important to note that genes do not work in isolation, but that they influence each other, increasing or decreasing the activity of other genes. Those interactions can be described as a complex network, called interactome, which is often used when doing RNA-seq analysis. Figure 3.3 shows an extract of the human interactome of various genes interacting.

The same figure also shows the complexity and high interconnectivity of the human interactome. When analysing gene interactions it is common to work only with a subset of the complete interactome, for example by starting from previously identified genes of interest and their neighbours.

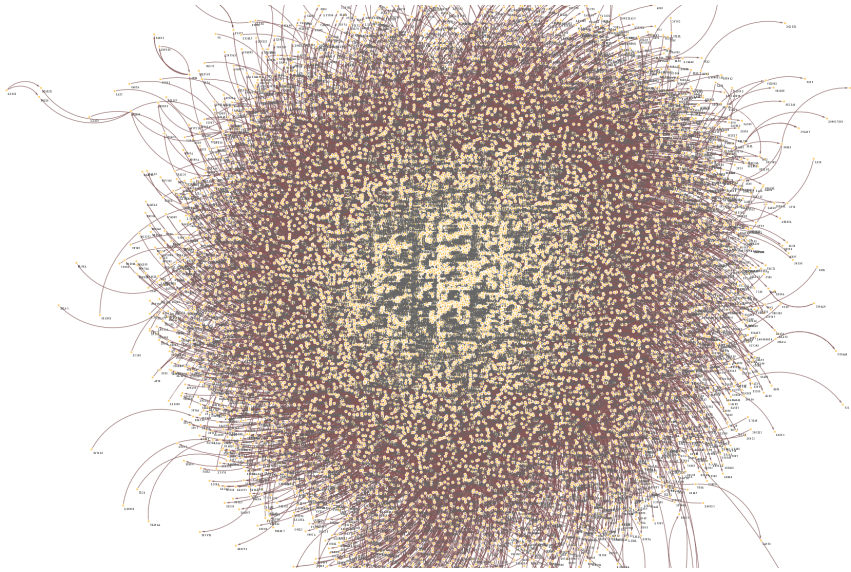


Fig. 3.3.: Extract of the human interactome provided by BioGrid [TBS⁺06] and visualized through Cytoscape

By studying gene expression levels across those networks, it is possible to study the effect a change in expression level of one gene on another gene. This allows for example to connect a gene not directly associated with a disease with a gene that is. Once the expression levels of a sample or the changes in expression levels between various samples are determined, those interactomes form a vital part of the data analysis process.

The interactions between the different genes can be based on various properties, all with a distinct use-case. Most commonly used and studied are interactomes based on protein to protein interactions (PPI), but other interactions exist, for example for regulatory effects between genes. Groups like BioGrid [TBS⁺06] or CCSB [RTC⁺14] provide interactomes of all known genes, whereas others, like KEGG [KSK⁺16] specialize on specific parts of those networks, called pathways.

Section 3.2.1 takes a look at the state of the art of the discussed use cases for RNA-seq.

3.2.1. State of the art

Many of the Transcriptomics data analysis steps when working with NGS data are very similar to the ones for genomics, as discussed in Section 3.1. This section will mainly discuss the data analysis steps which differ from genomics NGS data analysis. As discussed previously, during this thesis we focus on the quantification and gene expression analysis in the context of Transcriptomics. This is why the state of the art focuses on those aspects of the analysis.

As with genomics, the first step in the data analysis is the retrieval of the sequencing data. This sequencing data goes through a quality control and filtering step, which is essentially equal to the one discussed in Section 3.1.1. This step is followed by a sequence alignment step, which due to the nature of RNA-seq data slightly differs from DNA-seq, as discussed in Section 3.2.1. The proceeding quality control step of the aligned data is again mostly equal to the one seen in Section 3.1.1. This is followed by a quantification step which determines the expression levels of the different genes in the sample. The result of the quantification step is then used for expression analysis, comparing multiple samples and their gene expression

levels. Once the data analysis is done, it is common to visualize the data for verification or better understanding, a step which has some unique tools for transcriptomics.

The following sections will discuss the currently available tools and methods to perform those data analysis steps.

Alignment

To determine the source of the individual RNA sequences coming out of a sequencer the main method currently used is to align the sequences against a reference. While alignment-free solutions exist, such as Sailfish [PMK14], which use k-mer counting to identify the expressed transcripts, we focus on alignment based approaches.

There are two main approaches when aligning RNA-seq data against a reference, which differ in the choice of the reference that is used. RNA is encoded by DNA, for which in the case of the human genome we have a reference sequence. Most if not all expressed RNA can be found on the human reference sequence, which means, it can be aligned against the human reference sequence. Once mapped against the reference, the standard gene models (such as Gencode [HFG⁺12] or RefSeq [OWB⁺16]) make it possible to link the individual RNA sequences to the different genes. This allows for both the expression analysis of known genes, as well as the identification of novel genes.

The second approach is not to use the full genome reference, but the transcriptome, which is the sum of all known transcripts encoded by a particular genome, in our case the human genome. This second approach can be faster than aligning against the full reference genome, and avoids the problem of spliced alignment, which occurs when a sequence crosses the boundary between two exons. The downside of this approach is that the transcriptome reference limits the information that can be found, and it is harder to make a direct link between the DNA-seq and RNA-seq data.

To align the RNA-seq data, most if not all standard alignment algorithms can be used. In fact, bowtie [LTPS09] is commonly used to align RNA-seq data against a reference. Even though it does ungapped alignment, this does not cause a problem when aligning against a transcriptome reference.

But other more specialized aligners exist to map RNA-seq data against a full genome reference. Two of the most popular aligners are TopHat2 [KPT⁺13] and STAR [DDS⁺13], which both focus on spliced alignment of RNA-seq data. TopHat2 used Bowtie 2 as its backend for the actual alignment, splicing reads manually if required to get good alignment results. STAR on the other hand is a more integrated approach which does the alignment itself, resulting in a slightly better alignment quality and speed, but requires about 30 GB of RAM, more than most computers have today.

Quantification

Quantifying the sequences which align to a certain gene is important in RNA-seq data analysis to determine the activity of a gene a certain sample. Several tools exist to perform this task. One of the more commonly stand alone tools used for this task is HTSEQ [APH15], developed in python and made to work with all sorts of NGS data, including RNA-seq data. Cufflinks [TRGP12] is another popular tools, which is meant to be used alongside the Tophat 2 aligner. They form a toolsuite for RNA-seq gene expression analysis.

The last one is DEXseq [ARH12], which is an R [R D08] script which can be integrated into an R based RNA-seq data analysis workflow.

Expression analysis

After quantifying the gene expressions, this information is used for further analysis. The most common type of analysis is to compare the gene expression between multiple samples, also called differential gene expression analysis. This type of analysis allows to determine if any genes changed in expression level between multiple samples. There are many tools for this task, many of them custom made only used by a few. We are only going to present the most important ones.

Most tools to perform this analysis are R scripts which provide a set of tools inside R to facilitate the analysis. EdgeR [ZLR14] is one such example which is commonly used and constantly updated. The other example is DESeq 2 [LHA14], which has a similar goal to EdgeR and is widely used.

Once the differentially expressed genes are identified, the analysis is often continued with additional analysis of the results. Determining what genes are active or inactive is interesting in itself, but what is more often the question is if the genes identified are connected to a specific biological function.

An often used approach is to annotate the genes with the GO ontology and determine over or underrepresented biological functions. GoSeq [YWSO10] as well GoMiner [ZFW⁺03] are tools which can be used for that purpose.

For the prostate cancer data set, a P-value for differential expression between the treated and untreated cells was obtained for each gene using a Poisson exact test, equivalent to Fisher's exact test [15-17]. These P-values were then corrected for multiple testing [18] and the false discovery rate was set at 10⁻⁴.

Visualization

While RNA-seq data can be visualized with standard NGS genome browsers like IGV [TRM13], more specialized visualization exists.

Aside from using R based visualization (as many of the RNA-seq data analysis tools run inside the R environment), tools like RNASeqViewer [RZ14] exist. This RNA-seq visualization tool concentrates on the visualization of gene expression levels across genes and samples.

No RNA-seq data analysis visualization overview would be complete without Cytoscape [Chr05]. This locally installed Java applications has become the de facto standard for gene interaction graph visualization. Being constantly updated it is an essential tool to visualize the interactome which can be generated through RNA-seq analysis.

3.3. Epigenomics

Epigenomics, also called epigenetics, is an increasingly popular field in genetics promising to give new insights into gene regulation as well as how genetic information is inherited. In Epigenomics, the DNA is seen as more than a simple sequence of 4 nucleotides. There are different factors, outside of DNA, that influence the way the genome works and genes are expressed. We will focus on a process called methylation when looking at the field of epigenomics.

DNA methylation is a process in which a cytosine (C) nucleotide in the genome is modified through the addition of a small molecule from the methyl group. Figure 3.4 displays a

stretch of DNA with some methylated and some unmethylated cytosine nucleotides. The difference between the two forms is also shown on a chemical level. This modification does not alter the DNA sequence, the C remains a C and will also show up as such when using traditional DNA sequencing techniques. But it does alter how the DNA works, in particular gene regulation is influenced by the methylation of the promoter regions of a gene. To give an example how methylation influences an organism, we can look at ants and how the different roles inside an ant colony are handled. Inside an ant colony all ants descend from the same parents, the ant queen and the male ant that fertilized her. But even though all ants should have the same genetic makeup, they can develop in wildly different types of ants. As demonstrated by Chittka et al. [CWC12], ants specialize into different roles through methylation of certain genes, activating or deactivating them according to their role.

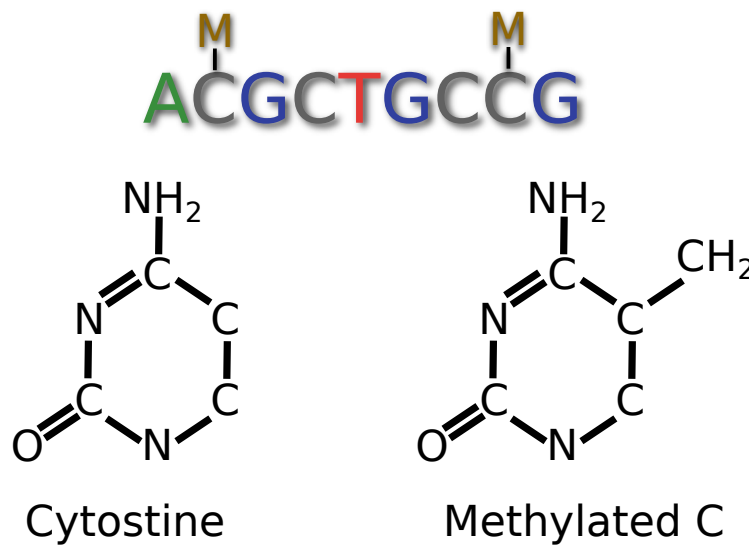


Fig. 3.4.: Methylation of the C nucleotide

Methylation is of course not only happening in ants, but also for cell differentiation as shown by Michalowsky and Jones [LP83]. Cell differentiation is the process used to specialize different cells inside the body into specific cell types, even though they all have the same DNA in them.

An important and not yet fully studied property of methylation is that it can be in part inherited and have an effect on the health of an individual. Heijmans et al. [HTS⁺08] studied the effect of the famine during the Dutch Hunger Winter of 1944-1945 on children that were exposed to this event prenatally. It could be demonstrated that the environment, especially during the crucial time of early development, has an influence on the epigenetics of an individual. Not only can the environment influence the epigenetics of an individual during its early development, but as shown by Ni et al. [NKC⁺16] those changes can be inherited over multiple generations.

While not all of those effects have been fully studied and understood, it adds a layer of complexity to the hereditary of genetic traits.

In humans (and other vertebrates) the location at which the methylation can occur is not completely random. In most cases, so called CpG locations can potentially be methylated. Those are locations in which a C nucleotide is directly followed by a G nucleotide. While the C nucleotide can be methylated at other locations, it is much less common.

The methylation can come in different forms, with 5-mC(5-methylcytosine) and 5-hmC(5-hydroxymethylcytosine) being two examples. Those different forms can have different roles in the functioning of the DNA as well as different sources of the methylation. The difference in effect and source of the different methylation types is still an open research question.

As seen in Section 3.2, studying the transcriptome and how genes are expressed is key to understanding the underlying mechanisms of genetic diseases. While genomics (Section 3.1) studies the modification of DNA to explain what is seen in the transcriptome, methylation analysis offers a different explanation for certain observations made in transcriptomics.

This shows how interconnected the different OMICs fields are, and how a single one of them does not allow to fully understand the complex mechanics working inside a cell.

As mentioned earlier, when using traditional DNA sequencing the methylated C is sequenced as a normal C and shows up with no modification. This is why when doing genomics analysis using sequencing data, methylation can be ignored. But there are techniques which make it possible to use next generation sequencing to perform methylation analysis. Methylated C nucleotides have a natural tendency to degenerate to the nucleotide T over time. With a chemical treatment of the DNA to be sequenced, this process can artificially be induced. During the so called bisulfite sequencing, the DNA is first treated with a bisulfite treatment to convert all methylated Cs to Ts. This conversion is valid for multiple types of methylation, including the mentioned 5-hmC [HPS⁺10]. Afterwards standard sequencing technologies are used, but with different downstream analysis tools.

Like in genomics, the sequenced bisulfite data can be aligned against a reference sequence. But bisulfite sequencing aware aligners have to be used for this task, as normal aligners will have trouble because of the Cs that have been converted to Ts.

The workflow for going from the raw sequencing data to a final analysis result has much in common with both genomics and transcriptomics. Figure 3.5 shows an overview of this process, which similarly is also described by Krueger et. al [KKFA12].

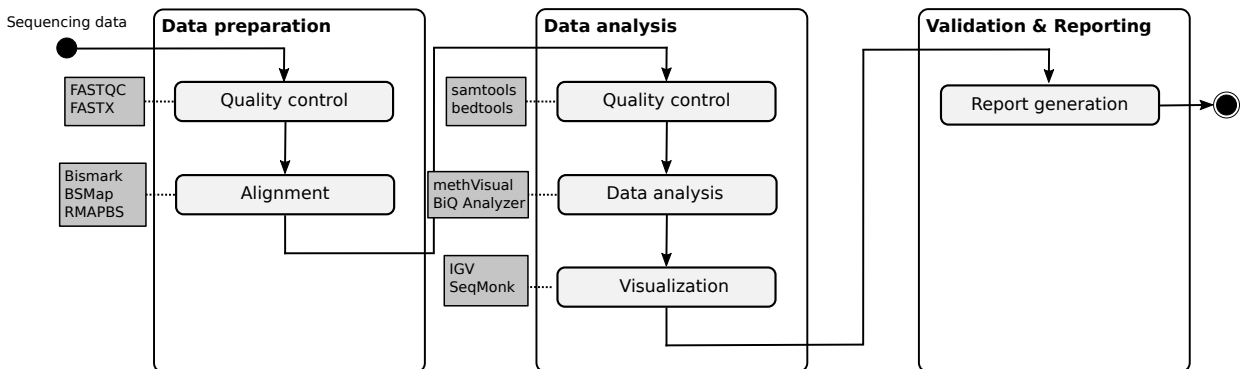


Fig. 3.5.: Epigenetics workflow separated into different steps. A collection of the available tools for every step are mentioned, based on author paper [WKDA15a]

In Section 3.3.1 we look at this process in more detail and specifically what tools exist to do it.

3.3.1. State of the art

Doing epigenomics data analysis, especially when looking at bisulfite sequencing data, has many things in common with genomics and transcriptomics data analysis. As seen in Figure 3.5, the general workflow has many similar steps as the two previously discussed NGS data

analysis types. While the treatment of the sample is different, in the end the original data-source for the data analysis is the output of an NGS sequencer. The first step of the data analysis starts again with the quality control, very similar to the one discussed in Section 3.1.1. This is followed by a sequence alignment step, which is particular in the case of bisulfite sequencing data due to the nature of the data, discussed in Section 3.3.1. After the alignment, the quality of the alignment is tested like with the alignments of the other data-sources (Section 3.1.1). The data is then analyzed to determine methylation levels of various genes, either inside of one sample or across multiple samples. This is discussed in Section 3.3.1. The results of the analysis is often visualized, just like with the other data-sources, to better understand as well as to validate the analysis results. This is done through genome browsers and similar applications, discussed in Section 3.3.1.

The following Sections discuss the tools and methods used to perform this type of data analysis. A more in-depth discussion of current methods can be found in [Boc12].

Alignment

Multiple sequence aligners exist which specialize in bisulfite sequencing data. Bisulfite sequencing data has the particularity that all non methylated C nucleotides are sequenced as T. This increases the difficulty to align them against the reference sequence, as normal NGS data aligners will treat those differences as variants.

An interesting comparison of different aligners which can handle bisulfite sequencing data can be found at [CSRM12]. Bismark [KA11] and BSMAP [XL09] are two of the most used aligners in the field.

Bismark is based on Bowtie[LTPS09] and aligns every read against 4 reference sequences. The normal forward and backward strands, as well as the modified strands where the C's are converted to Ts. This is done automatically through a python script, producing an alignment file as well as overall statistics of the aligned sample.

BSMAP is based on SOAP [LLKW08] and uses a slightly different strategy. Instead of being a wrapper around an existing aligner, it directly modifies the SOAP aligner to support bisulfite sequencing data.

Data analysis

Just like with RNA-seq data analysis, a lot of the data analysis of bisulfite sequencing data is done using the R environment. The first example is methVisual [ZS10], which provides a set of tools to analyze this type of data. But there are also standalone solutions, like BiQ Analyzer [BRM⁺05], which offer more complete approaches to analyze the data.

Visualization

Visualizing bisulfite sequencing data is in many ways similar to visualizing normal genomic NGS data. Because of this, most of the genome browsers mentioned in Section 3.1.1 can be used for this purpose. But because of the transformation of non-methylated C nucleotides to T nucleotides, many of the genome browsers will show a high amount of SNPs in the data. Very few genome browsers include the possibility to visualize bisulfite sequencing data correctly, notably IGV [TRM13] has a special bisulfite sequencing mode.

As an alternative, the tools presented in Section 3.3.1 provide also ways to visualize the data or at least the results of the data analysis.

3.4. File-formats

In this section we discuss the most important data-formats being used in NGS data analysis.

The way biological data is stored is crucial for its analysis. It not only determines how the data is accessed, but also how much disk space is used. Many different approaches exist to save specific data sets, but the domain bioinformatics has always favored the use of standard file-formats. Even if many custom solutions exist to solve various bioinformatics problems, certain file-formats became de facto standards. The following sections give an overview of those formats, organized by usage domains. Where possible, the described file-formats are also the ones used throughout this thesis.

Raw sequencing data

In NGS data analysis, the first step of data acquisition is usually to recover the data which comes out of a sequencer. The sequencing machines, presented in 2.1.2, output a list of sequences of different lengths which are stored in a digital file. Every sequence is composed out of the 4 basic nucleotides A,C,T and G. Commonly, every sequence is also attributed a quality score by the sequencer, indicating its degree of confidence in the particular nucleotide.

While there are multiple sequencers available, produced by different companies, there is a de facto standard in the domain of raw sequencing data. The FASTQ [CFG⁺09] file-format has become very popular and is supported by most bioinformatics tools which work with raw sequencing data. FASTQ is a text based file-format which stores for every sequence a name, the sequence of nucleotides and a quality information. The quality information is stored using a special value, called a *phred* score, which is a number between 0 and 60. A phred quality score can be converted into a probability (P) with the formula $P = 10^{-Q/10}$, where Q is the phred quality score and P the probability represented by it. For example a phred score of 10 represents a probability of 90% that a given nucleotide is the correct one.

Other file-formats for raw sequencing data exist, but are less commonly used. A multitude of tools exist to convert those file-formats into FASTQ files, which is often required to perform data analysis. One example of a different file-format is the combination of FASTA files with a separate quality file, both encoding the data as text files. The FASTA file-format was the original file-format to store sequencing data and it is highly similar to the FASTQ format, but lacking the quality information per nucleotide. Because of its lack of quality information it has been replaced by the FASTQ format, but it is still used to store reference sequence as seen in Section 3.4.

One other file-format worth mentioning is the *standard flowgram format*, in short SFF. This file-format encodes the sequencing data in a binary file, thus being more space efficient. It is mainly used by the Roche 454 sequencers.

Sequence alignment data

Sequence alignment data is a very important in many domains of NGS data analysis. This lead to a lot of standardization for this type of data, probably more than for any other file-format in bioinformatics. After aligning the raw sequencing data, sequence aligners output an alignment file. This alignment file is commonly saved as a BAM [LHW⁺09] file, or its textfile equivalent SAM. BAM files are the most used files to store alignment data. BAM and SAM files both contain the same information and follow the same format, but BAM

files are binary files using compression. This makes them much more space efficient than SAM files.

Recently another variation of the BAM/SAM file-format has emerged and slowly begins to be more important. The CRAM¹⁰ file-format is a compressed version of the BAM file-format, reducing its filesize notably by only storing the difference between the reference and the stored sequences.

Reading and writing all three file-formats is supported by the samtools [LHW⁺09] project, providing C and Java libraries.

Variants

Another important domain of storing NGS related data in files is the storing of variant information. While several dataformats exist, from custom text files to CSV files, one dataformat has become dominant in terms of storing variant information: the Variant Call Format (VCF).

The VCF file-format¹¹ has been developed among others by the 1000 genomes project. This very flexible dataformat allows to store all sorts of variant information, including the possibility to add custom annotations. There is a binary equivalent to VCF, which is a text file, called BCF, short for Binary VCF.

To also mention a new type of VCF, gVCF (which stands for genomic VCF), has been recently introduced. A gVCF file follows the VCF file-format, but instead of only including variants, it also includes the information about positions where no variant has been found. This can be useful for documentation purposes when the absence of a variant needs to be documented.

Annotations

When looking at NGS data, it is often important to have more information than just the sequence of DNA found in a certain sample. Other information, like the position of the genes, what parts of the genome are conserved between species or which parts of the genome are associated with certain diseases, is crucial when looking at the data. This additional information is generally grouped under the term, annotations.

Multiple dataformats have been developed over the years, all specialized in certain types of data to be displayed. Most of the file-formats are text files, such as the popular BED¹² and WIG¹³ files.

The BED file-format is a textual file-format used to store genomic regions. In the text file a series of regions, which are identified by their chromosome, start and end position, is described. They can have names and additional values (like color) associated with them.

The WIG file-format is more specialized in saving graph like data. It stores values for specific positions of the chromosome, allowing graphs to be drawn based on the data contained in a WIG file.

Both of those annotation files have multiple usages and are not only used when visualizing genomic data. They can be used to annotate genomic features, such as variants, or in the case of BED files, also to filter specific regions of a genome.

¹⁰<http://www.ebi.ac.uk/ena/software/cram-toolkit>

¹¹<https://vcftools.github.io/specs.html>

¹²<https://genome.ucsc.edu/FAQ/FAQformat.html#format1>

¹³<https://genome.ucsc.edu/goldenPath/help/wiggle.html>

Others

Many other data-formats related to NGS data analysis exist, but one of the most important, in the context of this thesis, is the FASTA file-format. FASTA, which stands for Fast-All, is a text file-format which is used to store biological sequences. Often they are DNA sequences, but can also be RNA, Proteins or others.

FASTA is today mainly used as the file-format to store reference sequences. The reference sequence, as explained in 3.1, contains the reference sequence of a certain species.

For completeness, it is to be mentioned that many custom file-formats exist in the world of bio-informatics data analysis. As most of the data that is analyzed, can often be described in a spread sheet like structure, it is no surprise that file-formats suited for this are often used. Both comma separated value (CSV) files as well as custom text files using tabs to separate the values are very commonly used, often lacking detailed documentation.

Data storage and compression

The size of NGS data keeps growing, because of faster sequencing technology and because sequencing becomes cheaper and thus can be performed more often. As many of the dataformats used in bioinformatics are text based, this quickly leads to problems in storing and accessing the data efficiently.

Different strategies exist to address this issue. One of the most common ones is to store the raw data files as gzip compressed files and to work directly with those files. This is notably used for raw sequencing data in the FASTQ format as well as Variants stored in the VCF file-format. Most bioinformatic tools support the use of gzip compressed files instead of the raw data files.

A different strategy is to use more specialized dataformat, which exploits the specificities of the data to be stored. One example of this is the CRAM file-format, mentioned in Section 3.4. Not only does it use compression to store the data, but it also reduces the filesize by only saving the difference between the aligned sequences and the reference sequence. As most sequences will have only a small differences with the reference, this leads to savings in file size. In the case of the CRAM file-format this can amount to about 30% filesize reduction compared to a standard compressed BAM file.

Similar strategies are used to compress raw data files commonly saved in FASTQ or gzipped FASTQ files. They either approach the problem by using specialized compression algorithms, fine tuned for DNA data, like MFCompress [PP13] and LFQC [NPR15]. Both of those approaches beat the standard compression used in bio-informatics, which is gzip.

Other approaches go further, by doing an initial alignments against a reference sequence to minimize the data to store, by only storing the difference to the reference for every sequence. This approach is used by LW-FQZip [ZLY⁺15], which achieves compression ratios of up to 0.15 for the raw data.

Even though there is a lot of research going in that area, no standard has been found which hinders the adoption of those methods of data compression.

3.5. Summary

In this chapter we discussed the different methods used for OMICS data analysis. The term OMICS regroups a large amount of different types of data-sources. We focused on three

OMICS data-sources, all three of which are using NGS technology for their analysis and are used in a diagnostics context. DNaseq, RNAseq and bisulfite sequencing, which all three offer a different but complementary view on the inner workings of a cell. For all three types of analyses we described a general workflow on how the data is analyzed and described the different methods used for the steps described in the workflows. Those methods are largely helped by a multitude of bioinformatics tools which automate the analysis of the RAW data.

Many of the mentioned methods are command line tools or lack the possibility to easily exchange their data with others, requiring an indepth technical knowledge from its users. This limits the amount of people that are able to autonomously perform NGS data analysis, which in turn limits progress in the field as well as how NGS data can be used in a diagnostics environment.

It is in the next Chapter 4 that we explore which of those tools are required in a diagnostic context, which is the main context in which the tools developed during this thesis are used. We also look at the three workflows described for the different OMICS data-sources and put them in a more abstract diagnostics context.

4. Diagnostics

This chapter relates the needs of diagnostics, in particular human diagnostics, to what modern genetics and especially NGS data analysis offer. It also shows the challenges faced today in diagnostics when working with NGS data and the solutions informatics and distributed processing can bring. The chapter starts with an introduction to what diagnostics is. This is followed by a description of what genetic disorders are and how they can be detected. In Section 4.3 we look at what software solutions need to improve the current situation in diagnostics, followed by a discussion of the chapter in Section 4.4.

4.1. Introduction

The word diagnostics comes from the word diagnosis which is defined as “the process of determining by examination the nature and circumstances of a diseased condition”¹. For the purpose of this thesis, the process of “examination” relates to the analysis of the DNA of the subject for which the condition needs to be determined.

When analyzing NGS data, it is common to search for either the reason why a patient has a certain illness or the risk that the patient or his offspring will develop such an illness. In diagnostics, the correct term used for illness is phenotype, as it is a more generic term than illness. A phenotype is an observable trait of an individual, which can be visual (like hair color), a behavior or a disease (like cancer), and thus applicable to more conditions than the term illness. Not all phenotypes are bad, as the example of hair color shows, and not all of them are caused by genetic variation. Linking the genetic information to those phenotypes is a major focus of research today in human genetics.

The challenge of linking phenotypes and genotypes on a given subject already starts with accurately describing the phenotype exhibited by the subject. While certain phenotypes may be easy to determine, like a missing limb, others, like the previously mentioned hair color, are much more complicated to classify. The near infinite amount of possible hair colors makes exact classification complicated. This is why projects like the Human Phenotype Ontology (HPO)[KDM⁺14] and the Online Mendelian Inheritance in Man (OMIM)² work on standard nomenclatures for phenotypes. This work helps to associate genetic changes with phenotypes systematically, making it easier to query them with automated tools. OMIM is an online database describing a vast amount of phenotypes and their relationship to genes, based on the original MIM (Mendelian Inheritance in Man) works started over 50 years ago. It is still widely used today and highly useful because of its use in many publications over the years. Due to its age, OMIM has various shortcomings, notably in regards to automatic data analysis using informatics tools. The HPO project tries to propose a solution for some of those shortcomings by developing an ontology which organizes the phenotypes in a tree structure, providing a hierarchy which can be used for advanced

¹<http://www.dictionary.com/browse/diagnosis> , status 6 sept. 2016

²<http://omim.org/>

analysis.

With the field of modern genetics being so young, the line between diagnostics and research is blurry at times. The vast amount of possible changes in the DNA for every individual makes it challenging to clearly determine if a patient has or doesn't have a specific condition. With the human genome being about three billion nucleotides long, most of the observed changes in an individual will be unique and thus have undocumented consequences. Different projects document variations already present in the general population, as well as their association with certain phenotypes and the frequency at which they are observed. The best known collection of variants comes from the dbSNP project, first published by Sherry et al. in 2001 [SWK⁺01] and which is actively updated regularly. As of the end of 2016, the version 147 dbSNP contains more than 153 million variants for the human genome ³. Another important source for diagnostics is the ClinVar [LLB⁺16] database, which focuses on a subset of variations in the human genome which are associated with certain phenotypes of clinical significance. Two other databases which are also often used in this context are the 1000 Genomes project [AAA⁺15] and Exac [Lek15] database. Both of databases offer the valuable information of the frequency at which various variants have been observed in the general population. This information can be used to identify variants common in the population and thus less likely to be causing the observed phenotype.

Much of diagnostics in genetics focuses on the identification of known variants associated with a certain disease. This is the easiest way to determine if an individual is affected by a certain genotype, because the particular change has already been observed and described in one or more other individuals. While this is the most optimal approach, often the changes found in an individual are unique. This is why in diagnostics it is studied if a gene which is associated with the phenotype in question contains any variants which have a prediction to impact the function of the gene in a negative way. The genes which are looked at for a specific genotype are often standardized to some extent in the form of gene panels. If no specific gene panel is specified for a given phenotype, resources like HPO and OMIM can help to determine the genes to analyze. This exact process shows the blurry line between clinical diagnostics and research. Once the initial clinical tests for known causes of a phenotype do not turn up any result, a clinical case can quickly become a research case. The implications of this are discussed by [Hul14] in a very interesting article.

Determining the impact of a variant on a gene can be more or less complicated depending on the nature of the change. A first relatively quick way to determine the approximate consequence of a variant is to determine where exactly the variant falls on a gene. Based on this information, tools like VEP (Variant effect predictor from Ensembl [CAB⁺15]) can determine if the protein encoded by the gene is potentially changed. This allows to categorize the variants into various groups like:

- *Synonymous*, no change in the encoded protein
- *Missense*, the encoded protein is changed
- *Stop codon addition/removal*, the encoded protein is shortened or extended
- *Intronic*, the variation falls inside an intron
- And many more

³http://www.ncbi.nlm.nih.gov/projects/SNP/snp_summary.cgi

As mentioned earlier, the line between diagnostics and research is often blurry, as very often the observed variations in a patient are unique and thus not yet documented. Looking only at the DNA gives a limited picture of what is happening inside the cells and the patient in general. This can lead to too many or too few potentially causative variants to be found in a sample. While this problem can be partially solved by also analyzing other family members of the patient, to filter out variants of not affected family members, not all cases can be solved with that approach.

This is why more and more, multiple OMICS data sources (see Chapter 3) are combined to better understand the consequences of the detected mutations. Having a tool which allows the geneticist to analyze and compare multiple OMICS data sources is important and a concept we presented at the European Human Genetics Conference during a poster presentation called *Towards integrative family analysis on OMICS data for individual patient diagnostics* [WKD14d].

While in the poster we focused on the combination of DNaseq, RNAseq and proteomics, this thesis focuses on DNaseq, RNAseq and Epigenetics as the OMICS sources. Variants detected in DNaseq can for example be validated through RNAseq which can show a modified splicing behavior of the gene in question. This allows geneticists to have a bigger picture than by just looking at one data source.

The study of bisulfite sequencing data in addition of RNAseq and DNaseq for a sample can help to identify the cause for certain phenotypes in cases where neither DNaseq nor RNAseq can reveal it. One example are tumor suppressor genes which can behave differently depending on their methylation, a line of study also discussed in this thesis (Section 9.1.4).

Combining those different OMICS data sources to fully understand the biological processes is increasingly important. While, which is why an integrated and standardized approach for their analysis is important to reduce the hurdles of analysis this type of data. Making the analysis of OMICS data easier than what is currently possible is also important in regards to personalized medicine and the increasing availability of NGS data. It is possible to imagine a world in which every person has been sequenced, and even the family doctor needs to access and interpret this data. This would allow them to propose individual treatment plans according to the genetic dispositions of their patients, without having to go through a geneticists, which could quickly become a bottleneck for the data analysis. An extreme case of this vision is that even the individual people can look and analyze their genome at home, if the usability and computational complexity of the data analysis can be fixed.

The next Section 4.2 looks into more detail what the actual genetic disorders are that can be diagnosed.

4.2. Genetic disorders

Genetic disorders are changes in the DNA that cause a certain phenotype which is not desired. Those genetic disorders can either be inherited by the parents, which may or may not be affected themselves, or the disorder is the result of a de-novo mutation in the subject. De-novo mutations are mutations which neither parent has, and can only be found in the offspring. On average, around 80 de-novo mutations can be found in every person [KFM⁺12], which they in turn pass down to their own offspring. Most of those changes are harmless and do not cause any significant change in the offspring. Yet some do have a significant effect and account for as much as 2/3 of the diagnoses made in genetics [YMR⁺13].

The exact location at which a mutation occurs determines its consequence. Most mutations fall outside the regions of any gene and have no consequence. But when a mutation falls inside a gene, it can modify the correct functioning of that gene. There are many ways a mutation inside a gene can alter its function. If the mutation falls inside the coding regions of a gene, the protein which is created by the gene can be altered. The coding regions of a gene are the regions that are translated into RNA which in turn is translated into a protein. As seen in Section 2.1.1, the protein created by the gene depends on the DNA sequence which is encoded through the codon table. Certain changes in the coding sequence will alter the function of the gene, which in the worst case can cause a complete loss of function of the protein. This can happen, if a stop codon is introduced inside the coding region, thus resulting in a much shorter and likely non functioning protein.

As explained previously, humans have 2 copies of every chromosome (except for males, a subject we will discuss later). Most mutations are only present in one of the chromosomes. Those mutations only present in one copy of the gene, are called heterozygous. The ones present in both copies of the gene are called homozygous. If a heterozygous variant has an effect on the gene function depends on the variant being recessive or dominant. If a variant is recessive, the affected copy of the gene does not impact the function of the gene, as the second copy still continues to work. A dominant variant on the other hand will cause the gene to cease working correctly even with just one copy being affected.

This plays a big role in hereditary diseases, where some mutations can have terrible consequences when homozygous, but none if heterozygous. As long as only one parent has the genetic defect, the child is unlikely to be affected as well, as a de-novo mutation in the child would be required for it to have a homozygous copy of that variation.

But this is not true for phenotypes caused by genes on the X and Y chromosomes, also called X-linked and Y-linked phenotypes. In humans, the X and Y chromosome are special cases as they do not always come in two copies like all other chromosomes. Individuals with two X chromosomes are females, and individuals with one copy of the X and one of the Y chromosome are males. This fact has consequences on the way variants on those two chromosomes are inherited. Recessive genes with a variant behave like genes on other chromosomes in females. In males on the other hand, as they only carry one copy of the X chromosome, variants on those genes are automatically “dominant”. This can lead to the well-known issue of color blindness, which affects men at about 6% and women at 0.4% of the general population[CGT14], but also other phenotypes such as hearing loss, a subject for which this thesis helped to find new knowledge [RBN⁺14], further discussed in Section 9.2.

Similar things apply to Y-linked phenotypes, which can only be inherited through the paternal line. Any phenotype linked to the Y chromosome cannot be inherited by a female offspring, but will be inherited by all male offspring. While very rare, a common Y-linked phenotype is male infertility (which of course is difficult to be inherited).

Last but not least we want to mention another type of genetic disorder, not directly linked to individual variants, but to much larger modifications of the genome. During cell division, many errors can happen that affect the copied genome, including the mentioned SNPs or small indels which change a small amount of nucleotides in the sequence. Another type of error that can happen are so called structural variations (SVs), which change the structure of the copied DNA. This means, large portions of a chromosome can be duplicated, inverted, deleted or even moved to another chromosome. Many of those events are very hard to detect using NGS, because of the limitation of the read lengths produced by those technologies. A

sequence originating from an inverted portion of the genome will for example align perfectly fine to a reference genome. Only sequences covering the borders of the inverted region will show unusual alignment behavior, usually resulting in their dismissal, an event that is hard to distinguish from the normal background noise in the data. Even though NGS data is not ideally suited for this type of analysis, it is increasingly used to detect copy number variations (CNVs), which are essentially duplications and deletions of part of the genome. Using techniques discussed in Section 6.7.6, CNVs are basically detected by comparing the coverage of certain regions in a sample, with one or more reference samples. Using statistical approaches, one can estimate the probability that a duplication or deletion event happened in this particular region.

An extreme case of those duplication events is the duplication of an entire chromosome. This leads to various problems for the individual, also known as trisomy, for which the duplication of the chromosome 21, also called down-syndrome, is the most well-known example.

After having looked at how different genetic disorders are caused, we will now look in the next Section 4.3 how computer science can make the work of detecting and analyzing those genetic disorders easier for the geneticist.

4.3. Software requirements

Computer science has introduced new tools and solutions in all possible domains of our daily lives. Medicine is no exception, with applications ranging from patient management, laboratory automation to assisted surgery and just to mention one of the author's works, *A WoT approach to eHealth: case study of a hospital laboratory alert escalation system* [RPW⁺12], the handling and dispatching of medical urgencies. Genetics and biology in general is no exception, with even a special research domain, bio-informatics, which has been named after it. The exact needs of diagnostics or to be specific, a genetic diagnostics lab, in terms of data analysis software are hard to define and highly subjective. This is why many of the described requirements are based on the needs of the laboratory of human genetics of the University of Würzburg, for which the development during this thesis has been mostly done.

The general workflow, for any of the three technologies (genomics, transcriptomics and epigenomics) looked at during this thesis, is shown in Figure 4.1.

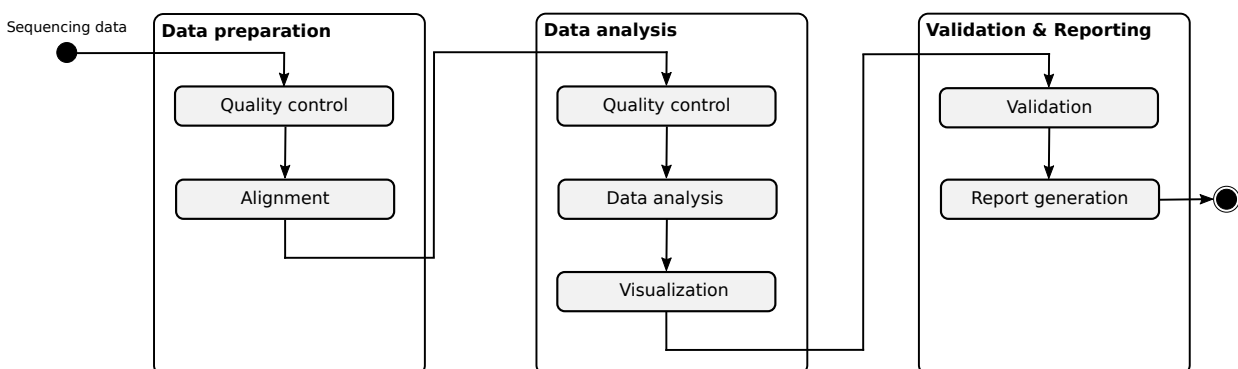


Fig. 4.1.: Generic UML activity diagram for NGS data analysis in the clinic.

The process of using and analyzing NGS data in the diagnostics, starts with the ac-

quisition of sequencing data coming from a next generation sequencer. Once this data is acquired, a quality control step ensures that the acquired data has sufficient quality for further analysis. Usually, during that quality control step, the data is also filtered to remove low quality sequences from the raw data. After verifying and correcting the quality of the raw data, the next data analysis step is, at least for the use-cases we try to cover in this thesis, sequence alignment. The sequence alignment step connects the raw data with the reference sequence, usually the human reference sequence, and provides the basic data source which is used during the actual data analysis.

After the sequence alignment step, another quality control step is common, to determine if the sequenced data aligned correctly and sufficiently to the reference. The most important quality control measure is the coverage analysis, which determines if the regions of interest in the sample to analyze are sufficiently covered.

It is after this second quality control step that the exact steps in the workflow can vary, depending on the technology used and the question asked. This is why we regroup this step under a general data analysis step. After the data analysis step, the results are often visualized to verify the validity of the data analysis. This is often done by opening the sequence alignment in a genome browser around the regions in which the data analysis identified a feature of interest.

Once the data analysis is done, it is common to verify the results using a second technology. One such example would be the usage of Sanger sequencing to validate the results of an NGS data analysis. Once the results have been finalized and verified, they are saved into a report which is handed of to a clinician, as well as archived.

The main requirements a diagnostics laboratory has for the software they use is that it facilitates the described workflow, mainly for genomics, but also for transcriptomics and epigenomics. The software should give the geneticists a certain independence from bio-informaticians for at least the most common data analysis use-cases. This is important, as most small laboratories do only have a limited amount of bio-informatics resources, making it important for the geneticists to be able to be independent for standard analyses. To enable this, the software should have an intuitive user interface, which integrates the various analysis tools and does not require command line tools to be executed manually by the user. The standard workflows should require little manual work by the user and still be flexible enough to allow exploratory analyses for more complex datasets. Existing infrastructures and workflows should also be able to be integrated into such a software. Examples of this are already existing sequencer installations which not only sequence but also align the data using sequencer optimized settings. Another example is the use of the existing infrastructure, which should optimally be used to reduce the need for infrastructure updates and expansions to a minimum. Last but not least, in the context of diagnostics the software should follow diagnostics software guidelines. This includes guidelines like the ACMG guidelines [RAB⁺15] which cover topics like how to classify variants and the HGVS nomenclature [DA00] which standardizes the naming of variants. Various local guidelines exist for various different countries as well as organizations inside those countries, all of which cannot be covered during this thesis. This is why we focus on traceability and reproducibility, two elements which are common in many guidelines.

4.4. Summary

In this chapter we discussed diagnostics in general and molecular diagnostics in particular. We introduced the different types of genetic disorders and how NGS data analysis can help to diagnose them. We also looked at the different initiatives which document known genetic changes to more easily analyze samples which have them, as well as to better understand samples with new ones. The fact that so much about DNA and how it affects an individual is still unknown, showed us the difficulty of clearly separating diagnostics and research. To better define the scope of the work of this thesis, we defined a general diagnostics workflow, based on the individual OMICS workflows presented in Chapter 3. We used this workflow to analyze the software needs of diagnosticians to follow that workflow in the most efficient way. The graphical NGS data analysis pipeline which was developed based on those needs is discussed in Chapter 6.

5. Parallel & distributed computing

In this chapter we give an overview of the field of parallel and distributed computing. We want to give the reader an understanding of the field and introduce the concepts used during works presented later. We also include an analysis of the state of the art, which is elaborated in further detail during later chapters where appropriate. To get a more in-depth understanding of the field, the reader is encouraged to read books like *Parallel Program Design: A Foundation* by M. Chandy [Cha88], *Introduction to Parallel Computing: Design and Analysis of Algorithms* by V. Kumar et. al [KGGK94], *Distributed Systems: Principles and Paradigms* by A. Tanenbaum et. al [TS06] or for a more recent one *Programming Distributed Computing Systems: A Foundational Approach* by C. Varela [Var13].

5.1. Introduction

Improving performance of computer algorithms is a constant concern of researchers and companies. One of the main techniques, especially today, is the use of parallel and distributed computing. Parallel and distributed computing are closely related concepts, both linked to concurrent computing. When a program can be split into different parts which can be executed at the same time, those parts are called *concurrent*. The different concurrent parts can be executed in any order, be it sequentially or in parallel, for example using multiple processors. What is important, is that the final result of the calculation is the same, no matter if the concurrent parts have been executed sequentially or in parallel.

The term *parallel computing* is used as soon as those concurrent parts are executed in parallel, meaning, at the same time. This can happen by using multiple processors on the same machine, or using multiple machines, in which case we speak of *distributed computing*.

Of course local and distributed parallelization use very different approaches to handle the parallelization. When parallelizing the concurrent parts on the same computer, traditionally *shared memory* is used for the communication between the different parts. For distributed computing, a different approach called *message passing* is used, in which messages are sent between the concurrent parts, but they can't directly access a shared memory.

Parallel computing, both local and/or distributed, can be used to significantly speed up the concurrent parts of an application, but does not help for the sequential parts of an application. This fact is also famously known as Amdahl's law [Amd67], which states that the speedup of an application using parallel computing is limited by its sequential parts. This law puts a limit to the performance increases which are possible through parallelization. If for example an algorithm has a sequential setup phase and concurrent calculation phase, even with the best speedup, the algorithm will never run faster than the sequential part.

While both local and distributed parallelization follow Amdahl's law that limits the maximum speedup, they both have unique sets of restrictions that limit their effectiveness. Local parallelization has both its strength and weakness in the shared memory that is used to communicate between the concurrent parts of the application. While the shared memory

allows for low latency and high bandwidth sharing of data, it is also limited through its bandwidth, which can lead to memory congestion if too many processors access the memory simultaneously. This problem is not present in the same way in distributed computing, as every machine handles its own local memory. On the other hand the communication overhead between the concurrent parts is much higher, both in terms of latency and bandwidth.

It is also important to note that parallelization, especially distributed computing, is not only used to speed up applications. Fault tolerance is an important aspect in today's computing world. Using distributed computing, it is possible to create fault tolerant applications where one more computer participating in the calculation can fail, but the application continues to work correctly.

We can see that parallel computing is a vast and complex topic, with the different approaches having their unique challenges and opportunities. This is why many different parallel computing models have been proposed over time to perform parallel computing. In the next Section we will describe those different approaches and how they evolved over time.

5.2. History

Here we describe how the field of computer science and especially parallel and distributed computing evolved over time. The focus is put on the parts of the development which are relevant to our works, as well as the time-line overlap with the development of genetics during the same time.

The concept of automating calculations for data analysis is not new. As early as in 1613 the word *computer* was used the first time to describe a person which performed calculations. For a long time people invented and used tools to reduce the complexity of this task, to only name the abacus (1100 BC) as one example of such a tool. It was in the early 19th century that the real practical proposals for computer like machines were developed. Notably Charles Babbage, helped by the famous Ada Lovelace, proposed two concepts to build an automated computing machine, sadly both were never finished because of monetary constraints. It was almost one century later, in 1936, that the first programmable computer named Z1 was developed. At the same time Alan Turing proposed the Turing machine, a now fundamental theoretical model of modern computers. During the next years and mainly motivated by the Second World War, computers were further developed and expanded their use cases. But they remained hard to use and required the space of entire rooms for a computing power that is magnitudes smaller than today's smartphones.

It was in 1953 that IBM released their first commercial computer, selling over 19k units even though it was only targeting the scientific community. During the same year, Watson and Crick discovered the double helix structure of the DNA. The development of computers rapidly progressed and they were used increasingly often in all sorts of domains. It was at this time, in 1965, that Gordon Moore made a prediction about the rate at which the computing power will evolve over time. While the so called Moore's Law was adapted a few times over the years, the current interpretation of a doubling of transistors on computing chips every 18 months remained remarkably precise.

Another interesting development in terms of distributed computing happened in 1968, only one year before the first manned moon landing. In that year, the ARPANET was

proposed, a network connecting multiple universities in the United States. The connection was made over phone lines, with a connection speed of 56Kb/s for the network backbone.

After the initial adoption of computers by the military, universities and the industry, the introduction of home computers started. It was during the mid to late 70s that the market of home computers started. With the Apple 1 in 1976, the Atari 400 in 1977 and the Commodore VC 20 in 1981, the usage of computers became accessible to a large part of the population about at the same time as DNA sequencing started. Also important to note is the release of the IBM Personal Computer Model 5150 in the same year, 1981, which also debuted the DOS operating System by Microsoft, which later evolved in the famous Windows operating system.

During this time, starting in the 70s, there was a divide in terms of processor technologies used in computers. While home computers used scalar processors which executed one operation on a data item at a time, scientific super computers used vector processors which were able to apply the same operation on multiple data items at a time. This type of parallelization to speed up computing was most famously used by the Cray super computers which stayed popular until the 90s.

Also in the early 70's, just after the introduction of ARPANET one famous distributed computing programming model was proposed. Carl Hewitt proposed the actor model, at a conference for artificial intelligence [HBS73]. One of the most influential implementations of the actor model was developed in 1986 at Ericsson through the programming language Erlang [CT09], which even today is source of inspiration for many actor model implementations.

Other models were introduced over time, such as the famous peer-to-peer (P2P) approach. This model became popular more for its use in file-sharing than to distribute calculation, notably with tools like Napster (1999) or BitTorrent (2001).

Around the same time, in 1999, the seti@home project introduced a new concept of distributed computing, by using the idle time of the computers of volunteers. The project analyzed telescope data in search of extraterrestrial life, a very time intensive task considering the amount of data to process. Later the project evolved into a more general project called BOINC, supporting researchers from all sorts of domains with their time intensive calculations.

That project is an example of the larger tendency that started around that time, which is now known as *Big Data*. Big Data science explores how to analyse very big, often unstructured data to extract knowledge. Many methods have been created to perform this type of analysis, which brings us to one of today's "hottest" distributed programming models which was introduced by Google in 2004. The MapReduce model [DG08] was developed to handle the challenges of big data and to easily scale over a large amount of computers with integrated fault tolerance.

But parallel computing is not limited to big clusters only accessibly by a small amount of people. During all that time, home computers continued to evolve but were mostly restricted to a single processor. It was around 2005 when CPU manufacturers like Intel and AMD released multi-core processors for the home market. Since then, multi-core systems started to dominate the computing landscape, to the point that today it is very rare to work with a system having only a single processing core. Even miniature open source computers like the raspberry pi zero ¹ now have at least two processor cores.

Slightly delayed but in parallel, a second type of parallel computing infrastructure was

¹<https://www.raspberrypi.org/products/pi-zero/>

deployed in most computers available today. Graphic cards, initially designed to accelerate the graphical calculations required for video games, evolved into multipurpose computing platforms, able to perform much more than only graphical calculations. Graphic card processors are made up of many small processing cores. While individually much less complex than a traditional processor, they are smaller and less complex which makes it possible to pack a great number of them on the same processor. NVIDIA released their programming language CUDA [NBGS08] in 2007, making it possible for programmers to offload certain calculations from the main processor to the GPU (Graphical processing unit). Because CUDA only runs on NVIDIA GPUs, in 2009 an open standard for calculations on GPUs was released, called OpenCL [SGS10]. Nowadays GPUs like the NVIDIA Tesla K80 contain 4992 cores, making it possible to perform massively parallel calculations.

The evolution of GPUs not only lead to much better looking computer games, but expanded their use to all sorts of domains. Be it for physics calculations, neural networks, weather forecasts or medical imaging, the use of those highly parallel co-processors has become very common.

This move from a single processor core to not only multi-core systems, but multi-core systems that are combined with GPUs, lead to heterogeneous parallel systems that are vastly different to program than the traditional computers. Combined with the ever increasing speed of networks connections, this lead to the development to a variety frameworks and tools being developed to allow programmers to program concurrent and distributed applications.

Another big computing model which has seen a big adoption over the last 10 years is cloud computing. While the concept was known for some time and in many aspects it is a way to commercialize existing grid infrastructures, it has seen a large adoption since its introduction around 2006 ². While software as a service (SaaS) was already a common business model, the cloud brought the concept of infrastructure as a service (IaaS) to the mainstream. In cloud computing the client can launch virtual machines that run the desired OS and code on a machine which corresponds to the specifications given by the client (with the upper limits set by the cloud provider). The cloud-provider transparently launches and manages those instances, with the client being able to launch or remove instances on the fly. The introduction of virtualization was a key component for the success of cloud computing, as it abstracts the underlying hardware from its users and at the same time allows for detailed control of the resources used. This computing model allowed researchers, companies and individuals to access relatively easily and cheaply the computing power they need, without having to worry about the infrastructure maintenance.

But this move to cloud computing brought its own set of challenges, notably in terms of data security. Especially when working with human DNA, which is a common use-case in this thesis, sending information over the internet, especially over country boundaries becomes a security risk. Some approaches exist to reduce this problem, such as private clouds, that reproduce the infrastructure of public clouds on the inside of an institution. One such private cloud is currently in planning at the University of Würzburg. Other ways are to completely anonymize all data sent over the network, but as shown [EN14], this is not sufficient to completely anonymize human DNA samples. Another, yet much more theoretical approach is the use of so called homomorphic encryption to secure data analysis in the cloud [MKA14]. This approach uses special encryption methods which make it possible to perform data analysis operations on encrypted data, without having to decrypt

²<http://www.google.com/press/podium/ses2006.html>

it in the process. The downsides are that currently the operations possible on encrypted data are still very limited and they are much more costly than the same operations on non-encrypted data, making the use of the cloud resources questionable. But even with those downsides, this approach is expected to greatly help genetics in the future [Tin15].

In the next Section 5.3 we present the various frameworks, tools or programming languages that enable programmers to use different techniques for distributed and parallel programming.

5.3. State of the art

This section gives an overview of the commonly used methods for parallel and distributed computing. The discussed concepts are not intended to be complete, but are meant to give an overview of the available models and techniques relevant to this thesis. The different approaches as well as their current state of the art are discussed.

We present three ways to develop parallel and distributed applications. In Section 5.3.1 present methods which allow parallelizing applications using one or multiple CPUs. Section 5.3.2 describes another local parallelization method, which uses GPUs to speed up calculations. Lastly, in Section 5.3.3 we describe the methods available to distribute calculations over multiple computers, be it in a grid, cloud or otherwise.

It is to note that the three domains can be combined, where a distributed application can use the multiple CPUs and GPUs of every machine which participates in the calculations. Some of the techniques presented, especially in the distributed category can as well be used to parallelize an application on CPU or even GPU, but is presented there as they are more commonly used there.

5.3.1. CPU

While not exhaustive, the following methods exist to parallelize applications on one machine using the CPU.

The first method is the use of threads, which are provided by most programming languages nowadays. Through the use of threads the programmers can split the normal application execution flow into multiple parallel execution flows. This method is heavily used but also arguably one of the toughest to master, as it puts a lot of responsibility onto the programmer. Splitting up an application in multiple threads and guarantee that the communication between them can easily result in race conditions and deadlocks which are hard to diagnose. On the other hand, for many applications the most efficient way to parallelize them is to use threads, which is not unlike the argument to use assembler code over compiled code for performance reasons.

A more automatic way to parallelize applications locally without having to manually split the execution into multiple threads has come under the form of OpenMP [DM98]. OpenMP allows the programmer to define certain parts of the application as parallel, and the compiler will figure out how to most efficiently parallelize it using threads. OpenMP has the advantage of coming as annotations to existing source code, which requires only minimal modification of existing applications to take advantage of it. It also allows the compilation of the application in either a multithreaded or sequential way, which helps development as well as understanding. The downside is that the use-cases in which it can

be used are rather limited, with OpenMP being mostly used to parallelize loops and not more complex parallel systems.

Also worth mentioning are the capabilities of modern CPUs for vector instructions. While earlier vector processors and scalar processors were separate concepts, today's CPUs integrate concepts from different models. This includes the ability to use vectorization instructions to apply the same operate in parallel on a vector of data. Different processor extension now coexist in modern processors, such as MMX, SSE and AVX. Examples of such operations would be the addition or multiplication of all numbers between two arrays. Instead of doing this for every number separately, the vectorization instructions can do this for multiple numbers in parallel and thus significantly speed up the computation for this type of operation. The programmer has a certain amount of control over the vectorization option, but can also leave it to the compiler to automatically vectorize the code where it is beneficial. Thanks to the lower complexity of vectorization operations compared to threads, the automatic vectorization found in modern compilers works very well.

The next Section 5.3.2 presents the methods to not only use locally available CPUs, but also GPUs which today contain a substantial amount of processing power.

5.3.2. GPGPU

GPGPU, short for "General Purpose Computation on Graphics Processing Unit", uses GPUs to perform computation. GPUs have a high amount of computing cores, which allows them to perform many concurrent calculations in parallel. While the individual processing core on a GPU is nowhere near as powerful as a CPU core, the immense number of them packed into a single chip makes them interesting to speed up calculations.

To program GPUs the most direct way is to use one of the GPU specific programming languages. The most known programming language for GPUs is CUDA [NBGS08] which was developed by NVIDIA and runs exclusively on NVIDIA GPUs. This proprietary programming language uses a C like syntax, allowing the programmer to define small kernels which are run on the individual GPU cores. CUDA is, despite its proprietary nature, widely used in the scientific community to speed up calculations of all sorts.

An open source alternative to CUDA also exists, OpenCL [SGS10], with the main advantage of running on the GPUs of all manufacturers and even on CPUs. At its core OpenCL is very similar to CUDA, although it often struggles to achieve the same performance due to missing optimizations by the compilers [KSA⁺10].

Using both CUDA and OpenCL requires a lot of knowledge from the programmer how to efficiently parallelize an application, not only generally speaking, but also to exploit the full potential of a particular GPU. In a similar fashion to OpenMP [DM98], OpenACC³ allows the programmer to define parallel regions (such as loops) in the code. Those are then automatically compiled into code that is able to run on GPUs, without the programmer having to program specifically for a CPU or GPU. Currently the OpenACC compilers are proprietary, which slows the adoption of the standard. But in recent version, the Gnu Compiler Collection (GCC) started to add basic support for OpenACC, potentially accelerating adoption once fully supported.

As an open source alternative to OpenACC, PPCG [VJC⁺13] has a similar approach, but is much more limited than OpenACC. As of late its development also slowed down, making it unlikely to catch up with its competition soon.

³<http://www.openacc.org/>

As we have briefly discussed, there are multiple ways to parallelize applications on one computer. The next Section 5.3.3 describes at the methods used when all the power of a single computer is not sufficient and the workload needs to be distributed over multiple machines.

5.3.3. Distributed computing

Various distributed computing models have been developed over the years. Nowadays most distributed computing models are based on the private memory approach instead of a distributed shared memory approach. To communicate, the different distributed instances use some sort of message passing, which can come in various forms. Especially in the high performance computing (HPC) world, MPI (Message passing interface) [For94] is currently the defacto standard approach to distribute calculations. MPI is a low level message passing API that enables developers to use message passing in C, C++ and Fortran applications, with Unofficial version existing for other languages like Java as well. Part of the reason of its success comes from the fact that the API is low level, which allows for a lot of manual optimizations to ideally utilize the available resources. Especially in an HPC environment this is important, but complicates the development of applications as it puts a lot of burden on the developer.

Before MPI existed, a similar model has been proposed under the form of the Actor model [HBS73]. In the actor model, algorithms are described as individual actors that are executed in parallel. Those actors communicate through messages, which prompt them to perform a certain task, which may again produce new messages. Most prominently Erlang [CT09] implements the actor model, but newer approaches like Akka ⁴ are now widely used.

A different take for distributed computing has been taken by the remote procedure call (RPC) approach. In RPC, function calls can be executed remotely, usually both synchronously or asynchronously. Calling methods, remotely or locally, is a well understood process that programmers are able to use. But simply being able to call remote function is not enough to create parallel distributed applications, which require additional tools, like threads and synchronization mechanisms. Examples of RPC implementations are CORBA and RMI, as well as the POP-Model, described at the end of this Section.

A more recent approach which gained a lot of traction is the MapReduce [DG08] computing model, most prominently implemented by Hadoop [Whi12]. The fundamental idea of MapReduce is to split an algorithm into two main parts, *map* and *reduce*. The first part, the mapping part, transforms a specific element into a different one, by mapping it to that new value. This new set of values is then sent to the reduce function, which collects all those values to reduce them to a single result. The idea behind this is that the individual parts can be programmed relatively easily, and more importantly, they can be distributed easily over many computers. Of course this approach also has downsides, like the restriction on which types of algorithms can be efficiently implemented using this type of abstraction. But the appeal of this new method is to easily scale up to a large amount of computers to perform the calculation, as well as having robust fault tolerance.

In this context it is also important to mention the POP model, which stands for Parallel Object Programming, was first proposed in 2002 [NK02] by Nguyen and Kuonen. The POP model has been implemented as an extension of the C++ programming language, called POP-C++ [NK07]. It has notably been used for projects like Alpine3D [KBL16], which

⁴<http://akka.io/>

is used to determine the risk of avalanches. Before the start of our works, a prototype of POP-Java [CS10] had been developed by Valentin Clement. This prototype will serve as the basis of further work and is described in more detail in Section 8.4.

The core idea of the POP model is that objects are ideal elements to distribute calculations: they can hold both data and methods and can communicate between each other. The concept of objects and how they interact is well understood by most programmers, which makes it an approach which can be easily understood by programmers. This goal is achieved by an RPC approach, not unlike CORBA, but with a more performance in mind, using a low overhead protocol. What distinguishes the POP model from other RPC approaches like CORBA and RMI, is the ability to easily create new objects remotely and the fact that it can also be used in a local environment to parallelize an algorithm over multiple processors. Thus, a major focus of the POP model is to allow for easy parallelization in a distributed as well as local system, abstracting most of the complexity from the programmer.

5.4. Summary

In this chapter we discussed distributed and parallel computing and how they evolved over the years. The different parallel and distributed programming models were presented and how they fit into the general history of the field. The different approaches to parallel computing on both CPUs and GPUs as well as distributed computing were discussed. A very diverse amount of methods exist to solve those problems, with sometimes very different concepts, making it hard for programmers to understand.

This is why we also presented the POP model, a distributed programming model intended to be more accessible to programmers than existing methods, due to its similarity to standard object oriented programming.

This is the end of the Foundations part and in the next part, Methods, we discuss the different contributions made during this thesis.

Part II.
Methods

6. Graphical pipeline

In this chapter we present our graphical NGS data analysis pipeline which intends to create an intuitive work environment for geneticists, especially in a diagnostic environment. The primary goal was to create a more accessible way to analyze NGS data, as well as to integrate the different methods described in Chapter 7. This chapter is partially based on the journal publication *DNaseq workflow in a diagnostic context, and an example of a user friendly implementation* [WKDA15a]. Real life examples of the usage of the work described in this chapter can be found in Chapter 9. The evaluation, done by a user survey, of the impact of the graphical data analysis pipeline is also presented in that chapter. Every aspect of the work presented in this chapter has been designed and implemented by the author, under the guidance of his supervisors.

6.1. Introduction

Analysing NGS data is a complex operation with many different tools that focus on one particular aspect or type of analysis. For many types of analyses, multiple alternative tools and methods exist to achieve the same or similar goal. We also explored the creation of NGS data analysis tools in a later chapter (Chapter 7), trying to address various issues in terms of the complexity of OMICS data analysis. While those works mainly focused on optimizing existing methods and data visualization, they did not address the complexity of handling those tools and doing the actually performing OMICS data analysis. Considering the multitude of tools, data formats and types of analyses that can be done, this is a critical aspect of NGS data analysis. Having an intuitive user interface which abstracts the underlying complexity of the task from the user, while still giving him enough flexibility to perform the work he wants to do, is crucial.

Various graphical or automatic OMICS data analysis tools exist, approaching this problem from different angles. An overview of those tools can be found in Section 3.1.1.

To understand the choices made that lead to the development of the graphical pipeline created during this thesis, the target audience of the application needs to be described in more detail. In the year 2011, NGS data analysis was still very new if not to say absent in the domain of diagnostics. The human genetics department of Würzburg started with NGS data analysis, mostly related to BRCA1 and BRCA2 cancer screening. But the solutions available at that time were lacking and too complex for geneticists not trained in bioinformatics to handle. This is why during the authors master thesis in computer science, a prototype of a more accessible friendly user interface for this type of analysis was created [Wol11]. Section 6.2 takes a closer look at this prototype that preceded this PhD thesis.

In Section 6.3 we look at the needs the human geneticists have and how those evolved since the initial prototype, as well as the implementation details and methods used to implement the graphical pipeline. The discussion of the chapter can be found in Section 6.8. Due to the nature of this chapter, the results are discussed in the Applications part of

the thesis, in Section 9 where the usage of developed pipeline in different research projects is detailed.

6.2. Prototype

The graphical pipeline presented in this chapter, GensearchNGS, initially started as a semester project during the master thesis of the author. The author developed a prototype of a genome browser during a semester project, under the supervision of the company Phenosystems SA. After this initial semester project, the genome browser was extended to create the first prototype of the graphical pipeline, called GensearchNGS. The goal was to create a graphical user interface to go from raw sequencing data to the final variant report that can be sent to the clinician. The previously developed genome browser was integrated for the data visualization part. This section gives a brief overview of the state of the project when this thesis started, more details can be found in [Wol11].

The basic motivation of the initial project was the lack of suitable tools for end users to perform NGS data analysis, especially in a diagnostic environment. The main use-case for the application was the Human genetics department at the University of Würzburg, which recently started to get NGS data, but lacked the tools to properly analyze the data. The focus of the initial prototype was rather narrow, focusing only on targeted gene sequencing (with the target being 1-2 genes per patient) and mostly Roche 454 sequencing data. Only the main workflow was implemented to show the viability of this approach. Even though the user had very little options, the main features already worked to support the initial needs of the Human genetics department in Würzburg. The raw sequencing data could be imported and managed for different patients. The data could also be aligned using either external aligners or a custom aligner developed during the project. Varscan 2 was used to scan the alignments for variants. While the variants could be visualized inside the genome browser, no filtering or annotation was available yet. But still, the user could go from the raw sequencing data to the visualization and report generation of manually validated variants. This already allowed the geneticists to work with their data, which at the time was limited in size, even if much manual work was required. Figure 6.1 shows the main window of the prototype application. As we will see later, the overall structure of the main window (as well as the genome browser) stayed similar. Figure 6.2 shows the prototype of the genome browser.

The prototype developed during the master thesis contained many of the core elements of the graphical pipeline developed later, but was lacking in terms of features and stability. On the other hand, the prototype could show that this type of approach is useful for the users and is as a good starting point. Over time, the lack of genome wide NGS analysis became apparent in all parts of the software. Much had to be rewritten and optimized to go from single gene datasets to full genome datasets. Indeed, scaling the prototype from single gene analysis to full genome analysis as well as greatly diversifying its features have been a big focus of the work done since the first prototype. At the same time, as the application became more and more popular and was no longer a pure prototype but actually used by geneticists for their daily work, a lot of focus has been put on polishing the user experience.

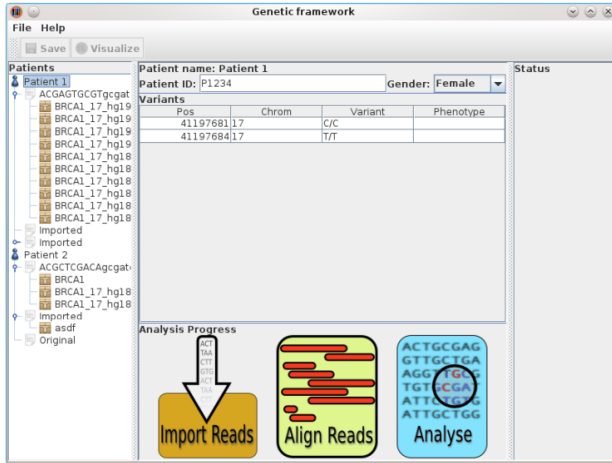


Fig. 6.1.: Prototype of the graphical pipeline [Wol11]

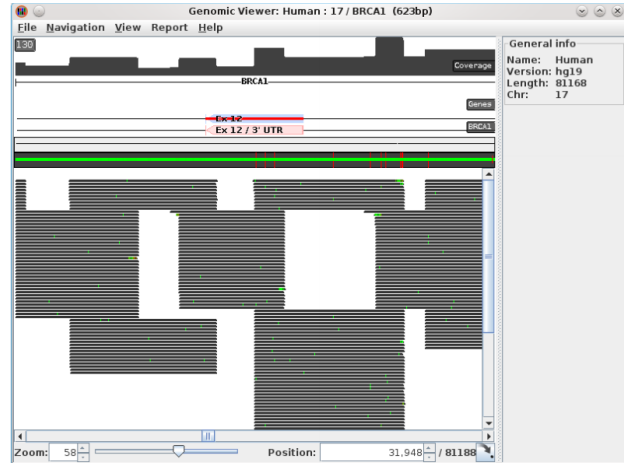


Fig. 6.2.: Prototype of the genome browser [Wol11]

6.3. Methods

To develop a user friendly graphical NGS data analysis pipeline, we used the prototype described in the previous Section 6.2 as a starting point. The development of the various features was guided by the needs of the Human genetics department at the University of Würzburg or other laboratories using the software. Through the different research projects which were performed in collaboration with the Human geneticists, many of which are described in more detail in Chapter 9, the feature set expanded.

The goal for the graphical pipeline can be stated as follows:

- User friendly NGS data analysis targeted at users with no bio-informatics skills
- Support data analysis tools required for human diagnostics, as well as basic research
- Integrate DNaseq, RNAseq and bisulfite sequencing analysis in the same application
- Run on standard hardware without the need for expensive computers
- Follow the guidelines required in a diagnostic setting

Those goals are modeled after the diagnostic software requirements described in Section 4.3 and especially the workflow described in Figure 4.1. Based on the described software requirements and the requirements of the human genetics department to perform both their clinical diagnostics and research tasks, we came up with the following list of features that are required. They closely follow the previously described workflows for DNA-seq (Section 3.1), RNA-seq(Section 3.2) and bisulfite sequencing (Section 3.3).

The core feature of the software, which was already prototyped in the first prototype, is the ability to manage different data analysis projects with their associated patients and data. This is described in more detail in Section 6.5. The different data analysis steps, such as quality control, sequence alignment and variant calling are described in Section 6.7. The analysis of the detected variants is discussed in Section 6.7.4, with more advanced variant analyses, like trio-analyses, described in Section 6.7.5. The detected and analysis of CNVs is described in Section 6.7.6, allowing to detect genomic features bigger than variants.

The way how the data analysis is parallelized and distributed over multiple computers is described in Section 6.7.7. Other features that are also part of the graphical pipeline, like the genome browser (Section 7.6), RNA-seq data analysis (Section 7.4) and methylation analysis (Section 7.5) are described in the data analysis Chapter 7. Those features are presented separately so that we can focus on the graphical pipeline itself in this chapter.

The graphical NGS data analysis pipeline containing the discussed feature has been developed in Java 6+ which allows it to run as a cross platform application on Windows, Linux and OSX. As will be detailed in the following chapters, we are able to work with various standard bioinformatics file formats (also described in Section 3.4). The support for those file formats for either read and/or write support has been done through both custom code and external libraries. The GUI was entirely developed using the Swing framework, which is part of Java, allowing to have a consistent user interface across all platforms.

The following external libraries have been used to develop various features in the pipeline: HTSJDK ¹ is used for all access, read or write, for BAM files, which store the alignment data. The library JTar 1.1 ² is used to access TAR archives, which are used for example to store to read archives including multiple VCF files. The GSON 2.2.4 ³ library is used when communicating with REST servers, such as the Ensembl REST API. For distribution purposes, both DIRMI ⁴ and POP-Java (Chapter 8) are used in the pipeline, especially for the aligner. We also include the bzip2, which comes from the Apache project, as well as Apache common-lang 3.3.2 and Apache commons-math 3.3, which are used for various purposes throughout the application.

For cloud computing support, we include JClouds 2.0 ⁵, which is also an Apache project.

6.4. User interface

The user interface has been designed to be as intuitive as possible. This was achieved through an iterative approach based on prototypes of features and user feedback. The feedback used to improve the interface came from multiple laboratories, with the major contributor being the laboratory of human genetics in the University of Würzburg.

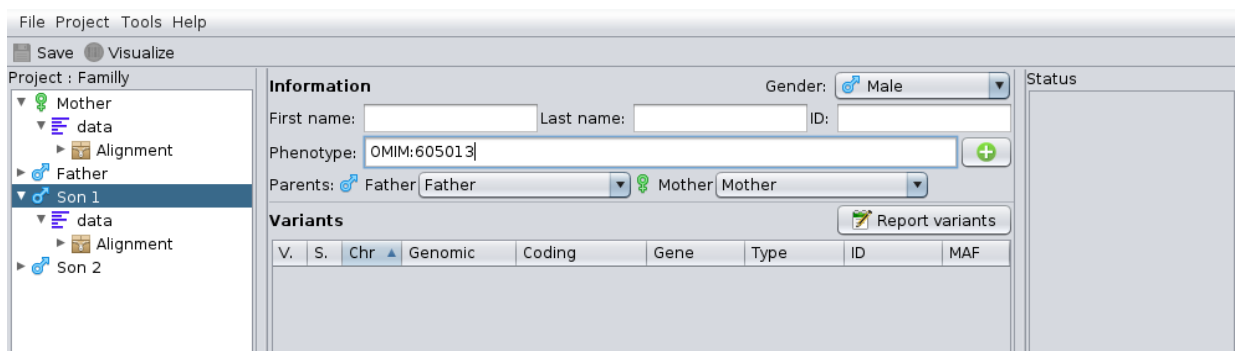


Fig. 6.3.: Main interface of the graphical pipeline

¹<http://samtools.github.io/htsjdk/>

²<https://github.com/kamranzafar/jtar>

³<https://github.com/google/gson>

⁴<https://github.com/cojen/Dirmi>

⁵<https://jclouds.apache.org/>

Figure 6.3 shows the main user interface of the graphical pipeline. On the left side, the user is presented with the various patients in the project, which have data files associated with them. The data is organized in a hierarchical fashion, meant to represent the general workflow of the sequencing data, going from raw data over an alignment to the final analysis report.

The context sensitive user interface displays only what is required by the user. This includes for example to only show RNA-seq specific options and tools when a user selects an RNA-seq dataset, to not overwhelm him with too many options that he often doesn't need.

In addition to this, as the application is used by different types of users in both diagnostics and research environments, we added the option to show or hide advanced options as well as experimental features. This includes options for the sequence aligners, variant caller as well as data analysis features which are under development but can be tested that way by a larger audience.

The various tools required for NGS data analysis are accessible either through the appropriate data entry in the data tree, or through the *Tools* menu.

6.5. Project management

When analyzing a lot of NGS data with many samples and various projects, it is important to have a comprehensive way to organize the data. Due to our focus on diagnostics and thus on patients, we opted for an approach in which projects and patients are the central units in which the data is organized.

The user can create independent projects which are self-contained. The project is saved as an entry point file, an associated folder and multiple configuration files which contain the configuration of the patients, references and regions of interest, all of which are explained later. The XML file format is used to store this data.

Inside a project, various patients can be created which have multiple attributes that can be modified by the user. Those include the name, the ID, gender and phenotype associated with this patient. Those attributes are used to identify the patient in the reports which are generated and also to influence data analysis as well as making it possible to use the search feature which finds a particular patient across multiple projects.

A project is not only defined by its patients. The user can also set various project specific settings, or the references to be used, which can vary from project to project. While in most cases the complete human reference genome is used, this is not always the case, and even if it is, the exact version which is used can vary (for example hg19 vs hg38). Figure 6.4 shows the interface provided to the user to manage the references for the current project.

This dialog includes the possibility to automatically download the human reference of a specific version, as well as individual gene references or import the already existing references from another project. This is done to decrease the manual work required by the user to a minimum, so that he does not have to download the reference manually from a website and also to make it easier for a laboratory to use a consistent set of references.

Another project specific setting the user can configure is shown in Figure 6.5. This dialog allows the user to configure the Regions of Interest, which are genomic regions which can be used as filters for the various data analysis steps, most notably, to filter variants. Those genomic regions are defined using the BED (see Section 3.4) file format. The BED file can either be directly provided by the user, or generated from a particular gene (or list

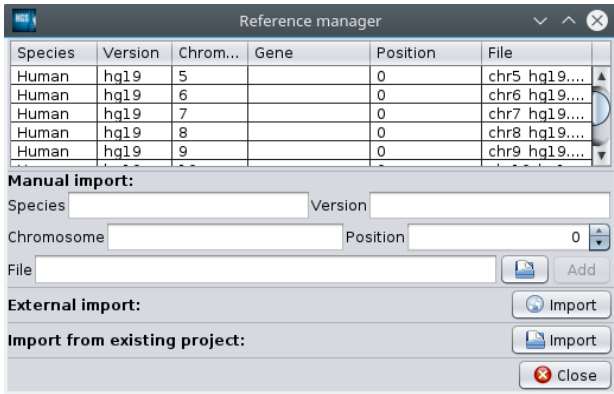


Fig. 6.4.: Reference management

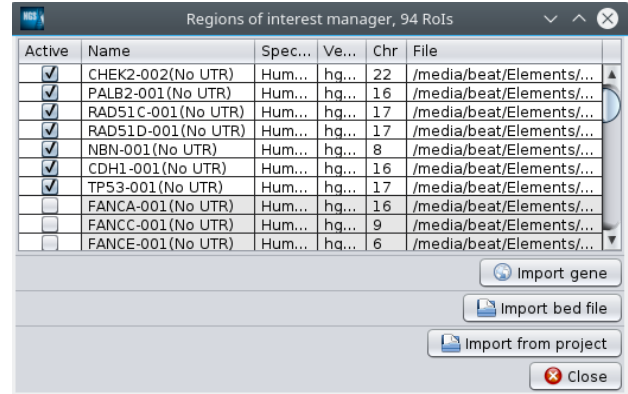


Fig. 6.5.: Regions of interest management

of genes) by using the gene information provided by Ensembl [CAB⁺15]. This feature has several use-cases, with most notably the ability for the user to specify the exact genes that he wants to analyze in this particular project. This comes in handy when analyzing a gene panel, using exome sequencing. Specifying the exact genes to analyze reduces the risk of incidental findings, something many laboratories are required to do when analyzing patient data.

Inside the project, the data is organized in a hierarchical fashion. This is also directly shown to the user, as seen in Figure 6.3, where we can see patients, which have associated raw data which in turn has an alignment. The way the data is organized is modeled after the general workflow for NGS data analysis described in Section 4.3 (see Figure 4.1) and intends to guide the user naturally through the data analysis.

A project has also project specific data analysis settings, which are updated and used automatically during the data analysis described in Section 6.7. Those settings include the default aligner used as well as its parameters, the filter settings for the variant lists and more. Like the other settings, they can be imported at any time from another project as well as being reused when creating a new project.

6.6. Annotations

Annotating genomic features, especially variants, is the core of much of the data analysis and interpretation that is done in NGS data analysis. We separate two types of annotations for the purpose of this section.

The first annotation type is the gene. Genes have a name and a specific position on a chromosome. They can in turn have multiple transcripts, each with a different function (such as protein coding, non-coding etc.). The human genome has an estimated 20'000-25'000 genes, with many not completely documented or not even discovered. Different organizations try to classify and name the known genes, to make it easier for researchers to work with them by providing them with a gene model. Two of the most used gene models are Gencode [HFG⁺12] and RefSeq [OWB⁺16], which in many parts are equal, but do have certain differences which can influence the data analysis results [FUR⁺15]. Knowing where the genes are is essential when annotating genomic data, as the genes are the currently understood functional units of the genome. Various types of information are associated with genes, beyond their basic type as stated earlier. While not provided by the

original gene model, additional data sources can be used to associate genes with relevant information. This can for example be the association of a gene with a specific phenotype, certain biological processes or its association with a pathway. Having a complete gene list, augmented with those additional information, allows any genomic feature with a known position on the genome to be associated with one multiple gene and thus connected with all information related to that gene.

The second type of annotation is feature specific, or as in our case, directly associated with certain variants. While a variant can be directly linked to genes and annotated that way, for many variants their effect has been studied in more detail. Projects like the 1000 Genomes project [AAA⁺15] and Exac [Lek15] provide information about the frequency at which a particular variant has been observed in the general population. Other projects like ClinVar [LLB⁺16] provide information about the pathogenicity of a variant, which is determined through previous studies that directly link a variant in an individual with its phenotype. But those annotations not only contain manually observed annotations, but also predictions about the effect of a particular variant. Various algorithms exist to that effect, such as Polyphen 2 [ASP⁺10], SIFT [KHN09] or the Human Splicing Finder [DHL⁺09].

Accessing and handling those different types of annotations, which come from various sources in different formats, can be challenging. This is why we designed a data structure for both genes and variants that can accommodate those different annotations and once loaded can be easily accessed by the application. The data structure has been designed in a way that multiple backends can be supported to retrieve the data. For our purposes, the main datasource is the Ensembl project [CAB⁺15] which provides a public API to access their gene model. The Ensembl gene model is based on the Gencode gene model. A separate proof of concept backend for the UCSC [KSF⁺02] was also developed, but is not actively used yet.

The Ensembl database is accessed both through its biomart webservice, as well as its REST webservice. The results of the webservice queries are cached locally, to allow faster access whenever the data is loaded a second time. The Ensembl webservice is used to get the general gene model, which contains all genes with their transcripts and exon locations. Those genes are then annotated using HPO [KDM⁺14] with phenotypes, Gene Ontology [BCD⁺15] for functional annotation and KEGG [KSK⁺16] to associate the genes with pathways. In addition, phenotypes are also loaded from Orphanet ⁶ and OMIM ⁷. For all those additional datasets, custom loaders have been developed to download and parse the annotation information which is then integrated into the internal gene data model used throughout the application. Both HPO and Gene Ontology use a standardized fileformat to describe their ontology, called the OBO file format, which not only contains the different annotations for the genes, but contains the connections between those different annotations in a tree like structure. This hierarchy allows of annotations allows for specialized or generic queries on the annotations. For example the GO term GO:0005102 (receptor binding) is a child of GO:0005515 (protein binding). When for example a gene is searched with a certain annotation and the query has no result, one can expand the search by moving one up in the hierarchy of annotations, launching a more generic search.

As for gene annotations, Ensembl regroups various data sources directly into its database, notably dbSNP [SWK⁺01], Exac, SIFT, Polyphen 2 and the 1000 genomes project. This allows us to query the Ensembl database about a particular variant and then annotate it

⁶<http://www.orpha.net/>

⁷<http://omim.org/>

with the information from various data sources, without having to access them one by one.

The internal data structure containing the annotations is used not only to annotate genomic features like variants, but also during visualization, to display exact gene information when viewing NGS data with the genome browser. How the annotations are used and in what context is discussed more in the next section about data analysis.

6.7. Data analysis

This section details the different types of data analyses that have been implemented inside the graphical pipeline. Some of the data analyses are described in more detail in Chapter 7, this is why this section concentrates on the user interface parts as well as the implementation on a non algorithmic level.

6.7.1. Quality control

Quality control is an important part of the NGS data analysis workflow. When importing raw data into the pipeline, the user is guided through a quality control and data conversion step. The standard raw data format which is supported inside the pipeline is the FASTQ format. This is mostly because most external tools, like aligners, expect FASTQ files as their input. As not all sequencers export their sequences in the FASTQ format, as a first step if the user inputs a different file format, it is converted to the FASTQ file-format. This is directly integrated into the pipeline, to not make it necessary for the user to use command line tools to convert the files. The following file-formats are supported and converted to FASTQ: standard flow format (SFF), FASTA and the 454 sequencers FASTA file accompanied by a quality file. Due to the modular nature of the code, other file-formats can easily be added.

After the conversion process finished, the resulting (or original) FASTQ file is analysed to determine the quality control values of the sample. At the same time, in the case of paired end sequencing, it is also verified that both supplied files match (as in, they have the same amount of sequences). The following metrics are calculated: Sequence length distribution, sequence quality distribution, sequence GC content distribution, the ACTG base distribution inside the reads, the quality per base, the amount of uncalled bases and the over-representation of k-mers of length 5 at the start and end of the reads. Figure 6.6 shows some of the generated statistics that are produced during this process.

The graphs shown in Figure 6.6 can be exported as images or a PDF file for archiving purposes as a quality control report.

Following the quality control step, the data can be filtered according to some given criteria. This includes the removal of barcodes, which can also be used to split the data into multiple samples. The user can also specify values to trim the data (for example to remove the last X bases of every sequence) or filter reads which are longer (or shorter) than a certain length.

The resulting raw data file can be saved for a given patient in the software, in order to align the data and to further analyze it.

6.7.2. Sequence alignment

Sequence alignment is one the most important tasks of the data analysis when analyzing NGS data. This important task is directly integrated into our graphical pipeline. As de-

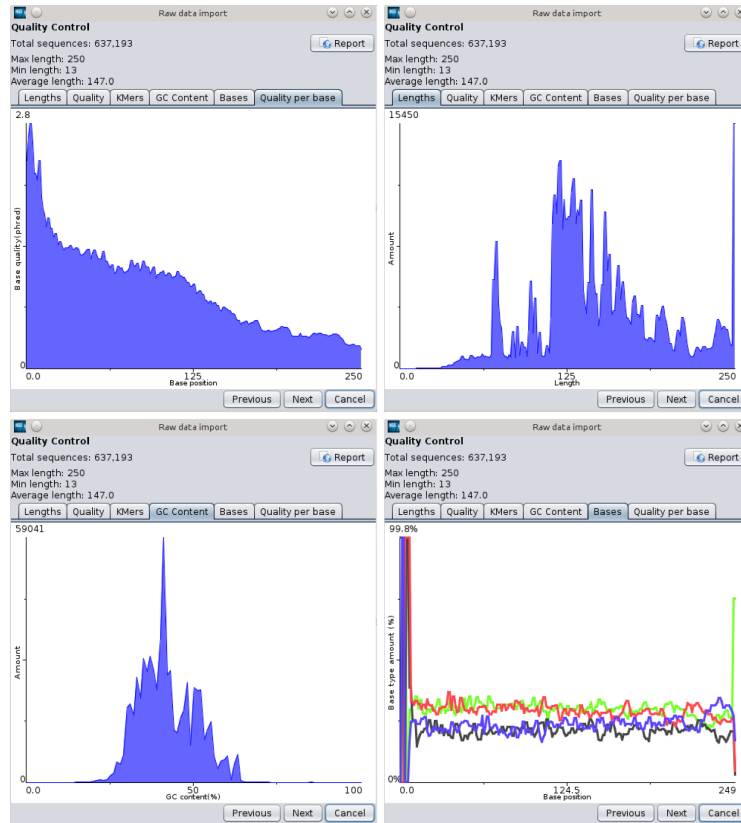


Fig. 6.6.: Overview of quality control measures for raw data

scribed earlier in Section 3.1.1, many sequence aligners currently exist and for reasons of cross platform compatibility, we also developed our own which is described in Section 7.1.

Each aligner has different advantages and disadvantages depending on the datasets used, and many laboratories have requirements for specific aligners. This is why we decided to integrate the sequence alignment in the form of *plugins*, allowing for multiple sequence aligners to be integrated. The main goal was to integrate them transparently to the user, without the user having to learn different workflows for each of them.

We identified the following general workflow that every aligner follows, and which can be abstracted to the user.

- Using one or two (paired end sequencing) FASTQ files as input.
- One more multiple FASTA files as the reference sequence input.
- Creation of a reusable reference index before the first alignment, stored alongside the reference sequence.
- Ability to set options that influence the alignment process.
- They create a non-sorted SAM file, which contains the alignment details of every sequence that was aligned.
- The conversion of that file to a sorted BAM file, in our case using the HTSJDK library

This last step is required as most NGS data analysis tools require sorted BAM files to perform their analysis. BAM files are a more space efficient version of SAM files, which can also be read faster.

We implemented plugins for all major aligners used by the laboratories with which we collaborate. Those are BWA [LD09] (and BWA-MEM [Li13]), Bowtie [LTSP09], Bowtie 2[LS12a], Stampy [LG11], CUSHAW2 [LS12b] as well as our own aligner (Section 7.1). The user can either configure the location of the executable of those aligners himself in the configuration of the software, or if it is placed in the local path, they will be detected automatically and made available.

With sequence alignment being the most time consuming task in terms of data analysis, we implemented a way for the user to distribute the alignment over multiple computers. More about the distribution features implemented can be found in the aligner Section 7.1 and the general distribution Section 6.7.7 for the graphical pipeline.

With the sequence alignment being so costly in terms of computing time, it is more and more common for the sequencing machines to include an aligner which is custom fit for the specific data they produce. In that case, the generated alignment files, usually in the BAM format, can be directly imported into our graphical pipeline without having to perform the sequence alignment. The import of the alignment files is made as easy as possible for the user, through several means. If necessary, the imported data is converted into the right format. This is most commonly the case if an unsorted SAM file is provided instead of a position sorted BAM file.

The BAM file is also automatically associated with the correct references, using the references configured for the project, while still giving the user the choice of the sequence to use.

Additionally, the type of the imported data is automatically determined based on the aligner used to create the alignment file. By default the imported data is set to be DNA-seq data, but if an aligner specific to RNA-seq or bisulfite sequencing is detected, the data type is changed. The currently supported aligners are: Bismark [KA11] and BSMAP [XL09] bisulfite sequencing data and STAR [DDS⁺13] for RNA-seq data.

After either importing a BAM file or aligning raw data, the quality of the alignment is determined, similar to the quality of the raw data discussed in Section 6.7. The next Section 6.7.3 looks at the tools which allow the user to perform this quality control.

6.7.3. Coverage analysis

In this section we look at the second round of quality control, which determines the quality of the aligned data. This is an additional quality control step, after the one described in Section 6.7.1 that looked at the raw data, which takes an alignment as its input. It is just as important as the first quality control, as having good quality raw data does not guarantee that the targeted areas of the genome have been sequenced correctly. Some parts might be missing or badly covered, something which will be detected during this phase.

To determine if all relevant parts of the genome have been sequenced correctly, we integrated our custom developed coverage analysis tool. While the overall coverage of the alignment is displayed to the user, this information is often of very little importance when analyzing gene panels or exome datasets. Using the regions of interest defined by the user, we determine the coverage of the regions that are of interest for the analysis that is being performed. Figure 6.7 shows the user interface for coverage analysis.

Rol	Name	Chr	Start	Stop	Aver...	Medi...	1	10	20	50	100	500
TruS...	chr1...	chr17	6010	6168	22	25	100%	98.7%	67.3%	0%	0%	0%
TruS...	chr1...	chr17	11205	11332	101	106	100%	100%	100%	100%	56.2%	0%
TruS...	chr1...	chr17	11871	11981	68	68	100%	100%	100%	94.6%	0%	0%
TruS...	chr1...	chr17	13920	13995	40	40	100%	100%	100%	3.9%	0%	0%
TruS...	chr1...	chr17	22327	22407	13	15	100%	81.5%	0%	0%	0%	0%
TruS...	chr1...	chr17	62293	63714	1	0	10.7%	8.6%	3.4%	0%	0%	0%
TruS...	chr1...	chr17	65444	65593	17	18	100%	99.3%	39.3%	0%	0%	0%
TruS...	chr1...	chr17	96901	97076	59	60	100%	100%	100%	72.7%	0%	0%
TruS...	chr1...	chr17	1315...	1316...	86	86	100%	100%	100%	100%	2.3%	0%
TruS...	chr1...	chr17	1692...	1693...	35	37	100%	100%	90.1%	0%	0%	0%
TruS...	chr1...	chr17	1710...	1712...	21	19	100%	98.6%	48.3%	0%	0%	0%
TruS...	chr1...	chr17	1772...	1773...	15	16	100%	78.9%	31.6%	0%	0%	0%
TruS...	chr1...	chr17	1835...	1837...	0	0	0%	0%	0%	0%	0%	0%
TruS...	chr1...	chr17	2025...	2025...	0	0	0%	0%	0%	0%	0%	0%
TruS...	chr1...	chr17	2601...	2604...	3	3	81.6%	0%	0%	0%	0%	0%

Fig. 6.7.: Detailed region of interest coverage statistics

The user can indicate the minimum coverage that is required for every region, highlight the regions which do not meet this requirement. The resulting analysis can be exported as a report to document the data analysis.

Once the coverage has been analyzed and all regions have sufficient coverage, the actual data analysis can start, which in the case of DNA-seq is variant analysis, presented in the next Section 6.7.4.

6.7.4. Variant analysis

Variant analysis is the main data analysis for which the pipeline has been originally developed. It starts with the detection of variants, which is done through GNATY (Section 7.3), the variant caller developed during this thesis. The user can scan for variants and set the variant calling parameters from inside the pipeline, with the last settings being applied by default to make the analysis easier. Alternatively, an external variant calling file (for example a VCF file) can be imported, thus allowing the integration into an existing data analysis infrastructure.

Once the variants have been detected, the main part of the data analysis is to determine the relevant variants for a particular sample. Figure 6.8 shows the window displaying all the variants of a particular sample.

As can be seen in the variant list, a lot more information is present than what is present in the underlying variant file (See Appendix C.1). The variants are annotated with various information to make their analysis easier and allow the user to make a better judgement about their importance.

The first annotation is the link between the variants and their affected genes. The annotations are applied using our annotation framework described in Section 6.6. The variants are associated with the genes that they affect (determined by their position) and thus benefit from the annotations of the genes. Additionally, the effect of every variant is prediction on the genes it affects, in a similar fashion to the variant effect predictor from Ensembl [MPR⁺10]. We developed our own variant effect predictor, as relying on an online service is too slow when dealing with thousands of variants. The user is always shown the worst

File Annotations View Tools

Legend Annotate online

Chr	Pos	Name	Quality	Freque...	Coverage	Reads (...)	Var bala...	Pos bala...	Genes	Type	Known	Prediction	MAF
1	1459269	C>T	98.2%	42.3%	71	30	57.9%	47.9%	ATAD3A	Stop gained	rs1434634...		0.000...
1	20501582	G>A	100%	35.2%	91	32	68.4%	82%	PLA2G2C	Stop gained	rs12139100		0.185
1	47080679	G>A	100%	47.2%	53	25	66.7%	39.5%	MOB3C MKNK1	Stop gained	rs6671527		0.384
1	115236...	G>A	99.9%	42.8%	152	65	62.5%	67%	AMPD1 RN7SL	Stop gained	rs17602729	Pathoge...	0.053
1	158576...	C>A	99.9%	43.7%	151	66	78.4%	65.9%	OR10Z1 SP7A	Stop gained	rs1489988...		0.011
1	161476...	C>T	92.8%	47.1%	461	217	82.4%	80.1%	FCGR2A	Stop gained	rs9427397		0.054
3	113955...	A>C	99.8%	38.7%	62	24	84.6%	55%	ZNF80 RP11-5	Stop gained	rs3732781		0.187
4	70462042	C>T	99.9%	45%	131	59	73.5%	79.5%	UGT2A2 UGT2	Stop gained	rs4148301	Important	0.076
5	134782...	T>A	100%	43.9%	98	43	59.3%	88.5%	C5orf20 TIFAB	Stop gained	rs12520799		0.389
6	29069299	C>T	97.9%	54.1%	148	80	95.1%	92.2%	OR2J1	Stop gained	rs2394517		0.386
7	64438667	G>A	98.8%	94.4%	18	17	88.9%	100%	ZNF117	Stop gained	rs1404453		0.113
9	91606787	G>T	99.9%	41.4%	29	12	50%	31.8%	C9orf47 S1PR	Stop gained	rs3722194...		
9	125391...	G>A	99.7%	36.4%	55	20	81.8%	77.4%	OR1B1 RP11-6	Stop gained	rs1476860		0.332
11	7712471	C>T	99.8%	49.2%	63	31	93.8%	80%	OVCH2	Stop gained	rs4509745		0.471
11	48266736	C>G	100%	42.2%	135	57	90%	95.7%	OR4X2	Stop gained	rs7120775		0.167
11	48286231	T>A	99.9%	44.3%	106	47	88%	89.3%	OR4X1	Stop gained	rs10838851		0.339
11	55371381	G>A	100%	41.5%	65	27	42.1%	71.1%	OR4C11	Stop gained	rs75423534		0.073
11	60265002	C>T	100%	44.4%	81	36	63.6%	55.8%	MS4A12	Stop gained	rs2298553		0.455
11	63057925	G>A	99.3%	48.6%	142	69	97.1%	100%	SLC22A10	Stop gained	rs1790218		0.469
12	55641255	C>T	99.9%	37.8%	45	17	70%	66.7%	OR6C74	Stop gained	rs4522268		0.226

3 filters

- Type equals Stop gained
- Frequency bigger than 30
- Var. balance bigger than 0.4

Report Close

Fig. 6.8.: Variant list with three active filters

consequence of the variant across all genes and transcripts that it affects. Using regions of interest (Section 6.5), the user can specify the particular transcripts and genes he wants to use for the effect prediction. Other than the gene and effect prediction, which can be done mostly using locally cached data, there are also variant specific annotations which are added. Those come automatically through the Ensembl biomart webservice, adding information like if a variant is already known in a public database and at what frequency. Additional information like the predicted consequence, using SIFT [KHN09] and Polyphen 2 [ASP⁺10] scores, as well as the clinical consequence as described by ClinVar[LLB⁺16], are recovered. Those are regrouped in the prediction column, always showing the worst prediction of the three data sources. Alternatively to the Ensembl biomart webservice, the user can also annotate the variants using a local variant file. This can for example be the VCF file from dbSNP [SWK⁺01] or the output file of the Alamut batch analysis. Thanks to this, the annotation of big datasets can be speed up and the user also controls the exact version of the annotations applied to his data.

Once the annotation process is complete, the user can search for the relevant variants in his sample. Being able to search and filter variants is important, as most of the thousands of variants in a sample are not relevant to the analysis. The common tools to perform this task are not interactive, they apply a certain set of filters on a list of variants to produce a new variants list. We opted for a more user-friendly approach, with filters that update the variant list on the fly, allowing the user to better understand their impact on the result. The resulting interactive variant filtering interface was presented at the *13th International Symposium on Mutation in the Genome: detection, genome sequencing & interpretation* as a poster presentation [WKDA15b].

The user is presented with the choice of various filters, allowing to filter the different aspects of a variant to reduce the amount of variants in the list. The following filters have been implemented:

Base filters

There are seven base filters that apply to the basic properties of the variants. *Position* filters the variants based on their position. *Frequency* uses the frequency at which the variant has been observed. With the *coverage* filter, the variants can be filtered based on their coverage, for example to filter out low coverage regions. The *quality* filter is used to filter variants based on their quality, which is determined by the variant caller. The *supporting reads* filter looks at the amount of times the variant has actually been seen in the alignment. To filter variants based on their balance, be it the balance of the variant itself or the general positional balance, the *position balance* and *variant balance* filters are provided.

Gene filters

The gene filters apply to the gene that is affected by the variant, five of them have been implemented. The first filter, *genes count* allows to filter the variants according to the amount of genes they affect. This is notably useful to filter out variants which affect no gene. The second is the *gene names* filter, which filters variants according to the names of the affected genes. The third filter, called *exon distance* filters variants according to their distance to the closest exon, allowing the user to look at exonic or intronic variants. The fourth filter, called *phenotype*, filters variants on the phenotypes that are associated with the genes they affect. The last filter, *function*, is similar to the *phenotype* filter, but filters on both pathways and gene ontology annotations. This is particularly useful for exome or genome wide analyses, that do not have a definitive gene panel for the analysis.

Annotation filters

The annotation filters apply to the annotations specific to every variant. Five such filters have been implemented. The first and most used filter is the *type* filter, which filters the variants based on their consequence (such as missense). The minor allele frequency *MAF* filter allows the filtering on the information of how frequently the variant has been found in the general population. With the *known* filter, the known status of the variants can be filtered, removing known or unknown variants. The *prediction* filter filters on the worst prediction the variant has. The last filter *local classification* looks if the variant has been previously seen in the local variant database and if yes with which type.

Other filters

The two last filters are a little special, as they do not fit into any of the other categories. The *file* based filter takes either a variant file or a bed file, and determines if the variant is present in the given file. If a variant file is given, the same variant needs to be present (or not present) in the given file. If a bed file is used, the variant needs to fall inside (or outside) a region of the bed file. The second filter is very similar, but uses the *regions of interest* set for the current project to do the filtering (which works the same way as the bed file filter).

All of those filters can be combined, as well as used multiple times. This is for example useful to search only for heterozygous variants, where the minimal frequency could be set to 25% and the maximal frequency to 75%.

The interactive filtering, which also runs on slower computers with even full genome datasets, greatly helps the data analysis. While for clinical diagnostics the filters are usually fixed to certain values (which are remembered), it gives the user enough flexibility to do more broad types of analyses.

The presented variant analysis is used to analyze one sample at a time. For more advanced types of analyses, such as a trio analysis, Section 6.7.5 presents the tools implemented in the visual pipeline for that purpose.

6.7.5. Variant comparator

The variant analysis described in Section 6.7.4 works great when looking at single samples. But for clinical analysis (as well as research), it is often required to compare multiple samples between each other to determine the relevant variants. A common example of this is the trio analysis, which compares the variants from both parents of an individual as well as the individual. One possible use-case for this type of analysis is to find the so called de-novo variants which are only present in the offspring and not in the parents. It has been shown [VB12], those variants have a significant contribution to genetic disorders.

Two solutions to perform this type of analysis have been implemented, both with a different focus in terms of flexibility and performance. Comparing potentially millions of variants over multiple patients is a memory and computing power intensive operation. Especially if filters are applied to the variants which require online annotation, the overall time of the data analysis can quickly become problematic.

For this reason, the first method to analyze multiple samples at the same time by comparing them, called “Variant comparator” was implemented. The user defines during a first step multiple groups of samples (which are picked directly from the available samples inside the currently open project) to compare them. A sample group can have multiple variant filters applied, to reduce the amount of variants inside the group. It is also possible to define if a variant group is negative or not, making it possible for example to subtract all variants from one group to the variants in the samples of the other group. The mentioned de-novo analysis can thus be achieved by using two groups, one with the offspring, and one negative group with the samples of both parents.

The comparison has been optimized to keep the number of variants stored simultaneously in memory as low as possible, while, at the same time, prioritizing less costly filters. If for example a sample group requires its variants to have at least a coverage of 30 bp, as well as being known in a public database, the variants are first filtered according to their coverage as this information is easily obtainable. In Figure 6.9 we can see the setup dialog of the variant comparator tool, which lets the user create multiple groups as well as define their content. Figure 6.10 shows the end result of that comparison, with a variant list that is filtered according to the user settings.

While the users of the variant comparator tool, and in particular the University of Würzburg, greatly appreciated this tool for both clinical diagnostics and research (See [RBN⁺14], also discussed in Section 9.2), it lacked the flexibility and ease of use of the other tools of our graphical pipeline. As most datasets analyzed are not full genome or even full exome datasets, a more interactive tool was designed and implemented. The successful user interface of the sample variant list described in Section 6.7.4 has been used as the inspiration for this new tool.

Displaying and filtering a list of variants to determine the variants of interest in a single sample has been intuitive for the users of the application. We extended this notion by allowing the user to open the variant lists of multiple samples at the same time, automatically grouping the variants over multiple samples. Without having to indicate the exact comparison and relationship between the samples beforehand, the user can have an immediate

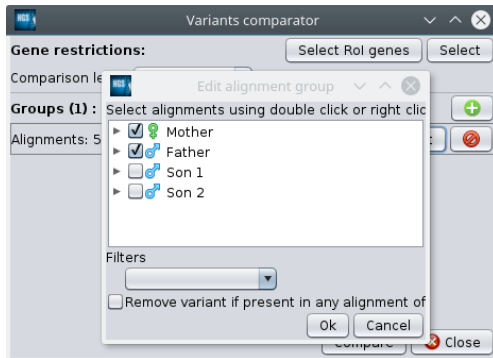


Fig. 6.9.: Variant comparator setup screen with group configuration

The screenshot shows the main window of the variant comparator tool. At the top, it says '46,556 variants on Mother, Father'. Below is a 'Legend' table with columns: Pos., Vari., Genes, Min..., Max..., Min..., Max..., Conseque..., Known, Pre..., MAF, On... The table lists various variants with their positions, types (e.g., A>G, T>C), genes (e.g., OR4F5, SAMD11), and predicted effects (e.g., Missense, Intrinsic v...). A 'Patient filtering' dialog is open over the table, showing filters for 'Father' (Heterozygous (< 75%)), 'Mother' (None), 'Son 1' (Present), and 'Son 2' (None). It also has fields for 'Min affected patients' (2) and 'Max affected patients' (4).

Fig. 6.10.: Result of the variant comparator tool

overview of the variants in multiple samples. We can see the merged variants window in Figure 6.11, with the patient specific filter dialog open.

The screenshot shows the 'Merged variant list of multiple samples' window. It displays a table with columns: Chr., Pos., Name, Father, Mother, Son_1, Son_2, Genes, Type, Known, Prediction, MAF. The table lists variants with their positions, names, and the percentage of affected individuals in each sample. A 'Patient filtering' dialog is open, showing filters for 'Father' (Heterozygous (< 75%)), 'Mother' (None), 'Son 1' (Present), and 'Son 2' (None). It also has fields for 'Min affected patients' (2) and 'Max affected patients' (4). Below the table, there are 'Filters' for 'Var. balance', 'Type', and 'Frequency'.

Fig. 6.11.: Merged variant list of multiple samples

In addition to allow for an easy and interactive way to find de-novo mutations in a sample (or many other more complicated tests), the tool can also be used to search for compound heterozygous mutations. Compound heterozygous variants are usually two variants coming as a pair on the same gene. The variants are both present in the parents, but only one in each parent. By inheriting both variants on the same gene, the combination can cause phenotypes not observed in either parent. Through the interactive filters provided to the users, finding those variants is straightforward. Especially compared to other tools which often require to use command line tools, our approach greatly reduces the complexity of

this task.

Although several tools exist to analyze the variants in a sample to find the phenotype causing variant, sometimes the observed phenotype does not come from a variant in the classical sense. So called copy number variations (CNV) which delete or duplicate complete regions in the genome can be at the source of those problems. To study them, we integrated a tool for CNV analysis, discussed in the next section 6.7.6.

6.7.6. Copy number variations

A big part of NGS data analysis is related to variants, also called SNVs, which stands for single nucleotide variations. But there are changes in a genome that can be larger than single nucleotide variations and they have a large impact on the function of the genome. Such changes, which are typically larger than 50 bp, are called structural variations. They can come in several forms, but we are mainly interested in copy number variations (CNVs), which represent a change in the amount of copies a certain region has.

To take an example, every exon of a gene is present two times, once on both chromosomes. If this exon is duplicated in the genome, it will be present 3 times in the genome, which can be seen as a 50% increase (on average) in coverage for that region. The same is true for deletions, where one or both copies of a specific region can be missing.

Traditionally, CNV analysis has been done using MLPA (Multiplex Ligation-dependent Probe Amplification) [SMW⁺02], which is a method which does not use NGS technologies. But recently, the usage of NGS data to perform CNV analysis has become increasingly popular. Various tools exist to perform this task, but most of them require command line tool experience and manual configuration of these tools. A great overview of the available tools is given by [ZWW⁺13].

To reduce the complexity of this type of analysis, we directly integrated into the graphical pipeline a tool to detect CNVs. The first use-case was to detect CNVs in a single gene, which lead to the publication of [BWO⁺15b], also discussed in Section 9.1.1. The user can select both the reference samples and the test sample to compare against the references, as well as set different analysis options. The results of the analysis can be seen in Figure 6.12.

The CNV analysis can run in two modes, the genome wide mode as well as a restricted mode. The initial development of the CNV tool only used the genome wide (or in that case gene wide) mode, which searched the complete alignment for CNVs. The user can set a minimal coverage, length as well as difference between the sample and the reference for a CNV to be detected. While this mode works well for small samples, as shown in [BWO⁺15b], it produces a lot of false positives when looking at large datasets.

This is why the second mode restricts the detection of CNVs to the regions of interest defined by the user. This is especially interesting for the case of exome sequencing, where the individual regions sequenced are exons. Using a BED file, usually provided by the sequencing kit, the CNV analysis is focused on the individual exons that have been sequenced. For every exon, the coverage for the reference and test samples are determined and compared using a t-test.

For normalization, we use the library size (amount of sequences in the aligned BAM file) of every sample. Alternatively, the mean and average coverage can be used, but they are prone to more false positives and are thus hidden by default. To calculate the fold change, by default the average of the reference and test samples is used, but alternatively the best fitting sample can be used.

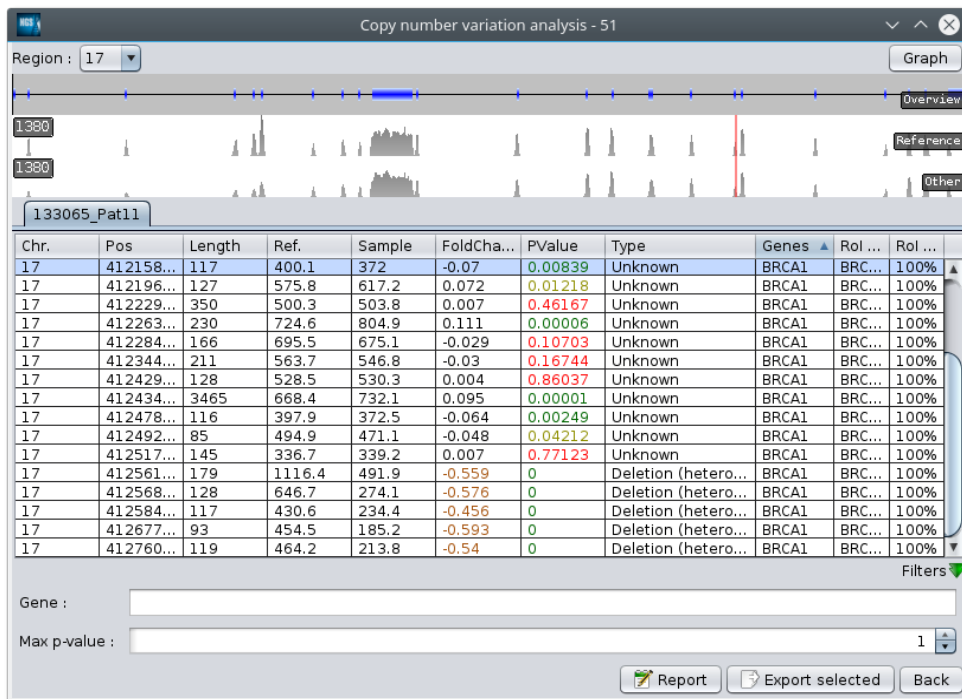


Fig. 6.12.: CNV analysis results

6.7.7. Distribution

The stated goals of the development of our graphical NGS data analysis pipeline is the optimal usage of existing infrastructure to reduce the need to invest in costly computing equipment. One of the most costly data analysis operations is the sequence alignment against a reference genome, which is part of the graphical pipeline and is discussed in Section 6.7.2. In Section 7.1.3 we describe how we distributed our own sequence aligner over multiple standard computers as well as on the cloud.

This functionality has been integrated directly in the pipeline, requiring little effort from the user to use it. The user has the choice of offering the resources of his local computer to other installations in the same network, as well as the possibility to use the resources of the other computers. This makes it possible to use the more powerful computers in a network during sequence alignment, while using the slower ones only as clients. During the start of the alignment process, the user can choose to use, or not, the other computers in the network. Additionally, he can choose not to align any data locally, but only on remote computers. Once the alignment process is started, an aligner job is automatically created on the remote computers without any additional actions needed from the user. The jobs are then finished and cleaned up as soon as the alignment is finished, but can also be manually canceled on the remote computer through the user interface. If any, or all, remote aligners are terminated, either willingly or through a malfunction, the alignment process continues as long as computer that started the alignment is still running. Any sequences sent to a remote computer which was shut down during the alignment are recovered. This type of fault tolerance assures that all sequences are aligned under any circumstance.

To detect the different computers on the same network we implemented a custom network discovery service. Every instance of the graphical pipeline that is setup to participate in network based alignment broadcasts its information through over the network using the

UDP protocol. Transparently without any user interaction the different installations of the graphical pipeline know about each other, making it possible to use them for distributed alignment.

If the locally available resources are not enough, we also integrated the possibility to use cloud resources from Amazon through the use of the JClouds library. To use the cloud service, the user only has to configure a user-name, password and user group specific to the amazon cloud to be able to use it. Once this information is set in the general configuration of the application, the usage of the cloud is transparent. Just like with the distribution over the computers available in the local network, the user can set the amount of cloud instances to be used at the start of the alignment. Starting up and destroying the virtual machine instances in the cloud is handled transparently.

Both approaches can also be combined, with both multiple computers in the same network as well as the cloud working together to align the data. Section 7.1.3 describes this process in more details.

6.8. Discussion

In this chapter we presented the development of a graphical NGS data analysis pipeline, aimed at supporting the analysis of OMICS data in a diagnostics and research environment. The feature set implemented has been strongly based on the requirement of the human genetics department at the University of Würzburg. The developed pipeline allows to perform the different NGS data analysis steps described in Chapter 4. By providing a locally installed graphical pipeline which creates a local cache of online resource which are used, we provide a stand-alone solution that can be used in an online or offline environment. This flexibility makes it possible to adapt to future requirements for NGS analysis, such as decentralized analysis or even analyzing NGS data at home.

The graphical pipeline presented is used by multiple laboratories and has supported research published in multiple works. The details of how the graphical pipeline has been used are presented in chapter 9, a chapter which also serves as a results chapter for the presented graphical pipeline.

The next chapter 7 presents the different analysis tools developed in the graphical pipeline in more detail.

7. Data analysis

This chapter presents the work that has been done during this thesis related to OMICS data analysis. This work is mostly centered around the creation of more efficient as well as more user-friendly implementations of OMICS data analysis tools. We look at different stand alone tools and methods developed during this thesis. Most of the tools discussed are integrated in a user-friendly graphical NGS data analysis pipeline, called GensearchNGS [WKDA15a], discussed in Chapter 6.

The chapter starts out by discussing sequence alignment (Section 7.1), the fundamental data analysis step for NGS data. The distribution of the workload over multiple computers and the cloud are discussed in Section 7.1.3. In Section 7.2 a new method called meta-alignment is presented. The method combines multiple sequence aligners to improve the alignment quality and makes the choice of the right aligner and alignment options easier. In Section 7.3 we look at a new tool developed for variant calling, which offers considerable speed improvements over current tools, making it possible to run the same analysis on cheaper computers. Next we look at RNAseq data analysis in Section 7.4, as well as methylation analysis in particular bisulfite sequencing data analysis in Section 7.5. To conclude this chapter, Section 7.6 discusses the genome browser we developed, which combines the visualization of DNaseq, RNAseq and bisulfite sequencing data. Finally, the results presented in this chapter are discussed in Section 7.7.

Unless specified, all the work in this chapter is the original work of the author.

7.1. Sequence alignment

In this section we discuss DNA sequence alignment, also called sequence mapping, a process crucial for NGS data analysis in diagnostics. We developed a sequence aligner to serve as the basis of other works, like the distribution of the alignment process over multiple computers or the cloud. We discuss the implementation of this sequence aligner and especially the distribution of its workload. The work presented in this chapter has been published in several papers, such as: “Distributed DNA alignment, a stream based approach” [Bea12], “A novel approach for heuristic pairwise DNA sequence alignment” [WK13a] and “Multilevel parallelism in sequence alignment using a streaming approach” [WMK15].

7.1.1. Introduction

Aligning sequences generated through NGS is one key aspect of DNA data analysis, especially in the domain of diagnostics. The other major method to analyze NGS data is sequence assembly, discussed in Chapter 3.1, a technique not used in our works. Sequence alignment can come in various forms, be it multiple sequence alignment or pairwise sequence alignment. In pairwise sequence alignment, two pieces of DNA are aligned against each other, to find the best way to combine them. Multiple sequence alignment on the other hand compares multiple sequences at the same time, for example from multiple species, to

align them against each other. This is for example useful to create phylogenetic trees, a type of analysis not used during this thesis. In this work we focused on pairwise sequence alignment, used to align the sequence data of one individual against a reference sequence.

Several generic and specialized sequence aligners exist today to align NGS data to a reference genome. We present some of the most used ones in Section 7.1.2. Even with the quality and variety of aligners already existing, we still required our own aligner given our requirements. The aligner, like the other methods described in the Methods chapter, is meant to serve a bigger picture and integrate into a graphical pipeline described in Chapter 6.

7.1.2. State of the art

This section gives an overview of the field of sequence alignment, especially on what algorithms are currently used. To find a list of commonly used aligners, refer to Section 3.1.1

Edit distance

The core part of sequence alignment against a reference, is to find the position on the reference at which the *edit distance* (see Section 3.1.1) between the sequence and the reference is the lowest. Various algorithms have been developed over the years to approach this problem. One of the first algorithms, which was used to align protein and nucleotide sequences, is the Needleman-Wunsch algorithm, published in 1970 [NW70]. This algorithm uses dynamic programming, a technique to optimize complex problems by dividing them into smaller, less complex problems. The results of those smaller problems are stored, so that whenever they are needed, they can quickly be retrieved. For example, the Needleman-Wunsch algorithm, works with one matrix of the size $M * N$, where M is the length of the sequence to align and N the length of the reference. After this matrix is filled (for the details we refer to the original paper), through a process called backtracking the optimal alignment is determined. This method used by the Needleman-Wunsch algorithm is able to align two sequences including gaps. The algorithm does so using a global alignment approach, which means, all characters in both sequences are aligned against each other as best as possible. This requires for both sequences to be ideally at a similar length, which is often the case when for example proteins are aligned.

Another popular alignment algorithm was proposed in 1981, called Smith-Waterman [SW81]. This local alignment algorithm searches for the best alignment between two sequences, but also considers subsequences, which is useful when the two sequences do not have the same length. Other than that, both algorithms (Needleman-Wunsch and Smith-Waterman) are actually very similar, with both being dynamic programming algorithms and very similar approaches to solve the problem. The main difference is, that in the Smith-Waterman algorithm, the matrix cannot contain negative scores, which allows for local alignments.

Both of the previously algorithms are widely used, but have one disadvantage in a biological context. Even though both of them support gapped alignment, (insertion and deletions are allowed), they have fixed costs for every gap, as well as gap extension. To give an example, an insertion of two nucleotides “costs” the same as two insertions of one nucleotide. In a biological context, this is not correct, as it is much more likely to have one longer insertion than multiple small ones. Algorithms which respect this property support the so

called *affine gaps*. Affine gaps define two different costs, one to open a gap, and one to extend a gap. One algorithm supporting this type of alignment is the Gotoh algorithm [Got82], which was published in 1982. The main difference of the Gotoh algorithm to the two previous algorithms is that three score matrixes instead of one are used, to track the scores of insertion and deletion separately.

All three of the mentioned algorithms, and variations of them, are commonly used in alignment algorithms. They are all rather slow algorithms, with execution time and memory complexities for all three of $O(mn)$, where m and n are the lengths of the two sequences to be aligned.

Reference indexes

Sequence alignment is a costly operation, especially because the search space is large. Millions of small sequences have to be aligned against a reference which is very long, in the case of the human genome 3 billion base pairs. The naive approach to align the sequences would be to test every sequence against every position on the reference. Considering the huge size of the reference, this is not doable in a reasonable time, which is why approaches to reduce the search space are required.

Because of this, sequence aligner use so called indexes, which are comparable to a book index. The goal of an index is to be able to quickly look up the potential places on the reference at which a sequence can be found.

The simplest index is the hash table indexing, which creates an index of all location of the k -mers in the reference sequence. A k -mer is a sequence of a fixed length. The index can quickly give the location of all occurrences for a given k -mer in the reference, comparable to an index in a book.

Another very popular index is the Burrows–Wheeler Transform (BWT), which is most notably used by the aligner BWA (Burrows–Wheeler Aligner) [LD09]. Originally, the Burrows–Wheeler Transform algorithm was invented for data compression (and is used by algorithms like bzip2¹). Later its usefulness as an index in sequence alignment has been discovered. Through backward search, the BWT reference index allows to perform a fast and memory efficient alignment against the reference sequence.

7.1.3. Methods

We developed a sequence aligner using the Java programming language which runs on all platforms supported by Java. As input files, FASTQ as well as gzipped FASTQ files are supported, for both paired end and single sequences. The reference sequence needs to be provided in the FASTA file format, either with all references in the same file, or one file per reference. The output format for the alignment is either a SAM or BAM file. You can find a description of the file formats in Section 3.4.

We used a stream based approach to implement the aligner, to split I/O operations from computationally intensive ones and make multithreading as well as distribution easier. Figure 7.1 shows the overall architecture of the algorithm.

We can see the main components of the aligner, which are the **Reader**, **Aligner** and **Writer**. The **Reader** is responsible to read the data to be aligned from the raw data files. Depending on the sequencing mode (paired end or not), one or two input files are read at

¹<http://bzip.org/>

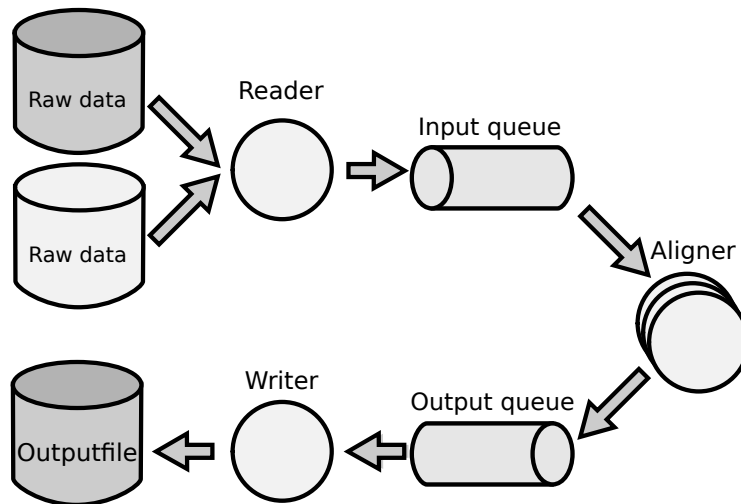


Fig. 7.1.: Sequence alignment algorithm architecture, modified from author paper [WMK15]

this stage of the alignment. To communicate, the different components of the algorithm are connected through queues (in our case an `ArrayBlockingQueue`). This allows every component to run in its own thread as well as to have the possibility to have multiple instances of the same components, notably the aligner component, run in parallel. Thus, the `Reader` component sends the data to align to the data `Input queue`. The `Aligner` component reads the raw data from that queue and aligns it against the reference. Once the alignment is done, the `Aligner` component puts the aligned sequences on the `Output queue`, which is read by the `Writer` component. This component saves the aligned sequences in an alignment file.

This stream based approach gives us a lot of flexibility in terms of scaling to more processors or even multiple machines. Section 7.1.3 will expand on how this architecture is used to distribute the workload over multiple computers and the cloud.

We will now look in more detail at the sequence alignment algorithm itself, which is the algorithm that runs in the `Aligner` component. The sequence alignment algorithm can be roughly divided into 4 steps that bring a raw sequence to its alignment. We can see in Figure 7.2 those different steps.

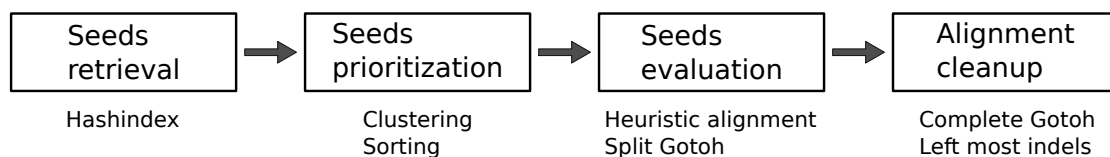


Fig. 7.2.: Sequence alignment algorithm overview, splitting the alignment process into four principal steps

In the first alignment step, the potential positions on the reference, called seeds, at which the sequence can be aligned are determined. We use a hash table index to perform this step. By default, the k -mer size used for the hash table index is 12, which provides a good compromise between memory size (1GB for the index) and uniqueness for every k -mer. At this stage, the sequence to align is split into all possible k -mers it contains, as well as their reverse complement (see Figure 7.3).

Looking up the k -mers in the reference index is a time critical step, which requires to

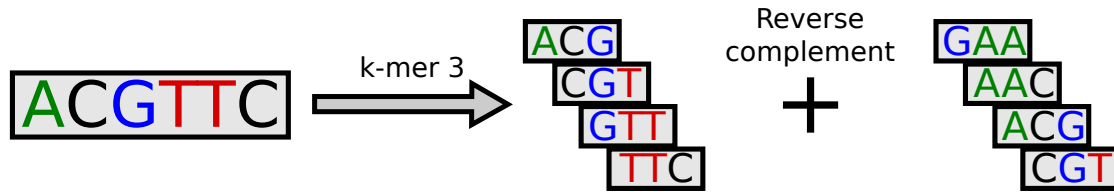


Fig. 7.3.: A sequence is split into its k-mers (size 3 in this example) and their reverse complement. Those k-mers are the seeds that are used to search the reference index for possible alignment positions.

create a hash for every k-mer (the same which is used to create the hash table index). We transform the ASCII value of the 4 nucleotides (1000001), C (1000011), T (1010100) and G (1000111) to values between 0 and 3 using the following formula 7.1:

$$V = (A \wedge 6) \gg 1 \quad (7.1)$$

V is the ID of every nucleotide and A is the ASCII value for every nucleotide. This formula uses the bits at position 2 and 3 from the right to create the unique ID for every nucleotide, something which can be done very fast.

Using the hash for the individual k-mers, the hash table index returns all their positions reference, split into the different sub references (usually by chromosome). This operation can result in a big amount seeds at which a sequence could be aligned against.

During the second step, we prioritize the seeds, to first align against the most promising ones, hopefully reducing the search space. To do this, the following metrics are determined to give a score to every seed:

- Number of directly adjacent seeds (which expand the seed with no break)
- Number of neighbors (amount of seeds at most $2 \times$ sequence length away)
- Uniqueness (how many times was this seed seen on the reference)

To determine the amount of neighbors and number of adjacent seeds in an optimal way, the seeds are first sorted according to their position on the reference. We use a custom version of the Timsort algorithm, which was developed by Tim Peters in 2002, to sort the seeds in $O(n \times \log(n))$ for the worst and average case and $O(n)$ in the best case. This step of the sequence alignment is performance critical, as every one of the millions of reads can have hundreds or thousands of seeds to sort. Once the seeds are sorted according to their position, we can cheaply determine their close and adjacent neighbors.

After having sorted the seeds according to their score, we evaluate them to determine if they point to a valid alignment. The evaluation of every seed is performed in two stages. At first, a quick heuristic is used to calculate an approximate alignment for the sequence at a given seed position. The dynamic programming algorithms described in Section 7.1.2 are very costly and are not practical to be applied to every candidate seed. Therefore we use a custom heuristic algorithm, described and published in [WK13a], which allows us to quickly determine if a position contains a possible alignment. This algorithm is between 6 to 24 times faster, depending on the length of the sequence to be aligned, than standard dynamic programming algorithms.

Once the viability of the seed has been evaluated, we use a custom version of the Gotoh [Got82] algorithm, to create an alignment for this position. In this phase, we use a split

Gotoh approach, where the left part and the right part of the sequence (split at the seed), are aligned separately. This greatly reduces the time required to perform the alignment. For every alignment, a score can be calculated, which is based on the amount of matches, mismatches and indels. Given this score, the best alignment for a given sequence can be determined when evaluating multiple seeds.

The last phase, which is a cleanup phase. This last phase cleans up the alignment of a sequence, to assure that multiple sequences aligned at the same place, are aligned the same way. First, all sequences are aligned using the complete Gotoh algorithm. This removes differences in alignment that could arise when two sequences are split at different places when performing the prior split Gotoh alignment. After this the CIGAR string, which determines at which location of the sequence indels are present as well as matches or mismatches, is improved and normalized. This normalization includes merging of indels (if possible), moving indels to the left most position (Figure 7.4) in the alignment as well as skipping faulty ends of the sequence (Figure 7.5).

Reference **ACGTTGCA**
 Alignment **GT - G**
 ↓
 Cleanup **G - TG**

Fig. 7.4.: Left most indel cleanup

Reference **ACGTTGCA**
 Alignment **ATTG**
 ↓
 Cleanup **~~A~~TTG**

Fig. 7.5.: Faulty end skipping cleanup

Several parameters can be set to influence the alignment process. Here is the list of parameters that are implemented in the algorithm:

- Allowed error rate (%) : The amount of errors in the alignment for it to stay valid (default 6%)
- Maximum indel length : The maximum length of indels (default 12)
- Tolerate more/longer indels : Less strict enforcement of the maximum indel length rule
- Try out all possible alignments: Evaluate every seed, greatly slows down alignment
- Sequences to align : Amount of sequences to align, mostly used for debugging

The next Section 7.1.3 discusses how this aligner has been distributed over multiple computers using different technologies.

Distributed alignment

Sequence alignment is a time critical process, it represents one of, if not the biggest task when doing NGS data analysis. With one of the goals being to run NGS data analysis on off the shelf desktop hardware, we expanded the presented aligner to not only use multi-core systems, but also run on multiple computers as well as on the cloud.

Figure 7.1 shows the general architecture of the aligner. We can see the **Aligner** component, which can be instantiated multiple times, in separate threads, to allow multi-threaded alignment. The interface between the **Aligner** component and the rest of the system is relatively simple, thanks to our stream processing based approach. The **Alignment** object takes unaligned sequences (or sequence pairs in the case of paired end sequencing) from the input

queue, and outputs the aligned sequences to the output queue. This can be used to run different implementations of the **Aligner** component, for example to offload the alignment to a remote computer. The distributed **Aligner** component can run alongside the local **Aligner** component, allowing for both distributed and local alignment at the same time.

This is the base idea we used to implement distributed alignment on top of the existing multi-core aligner. Figure 7.6 shows the modified aspect of the general architecture.

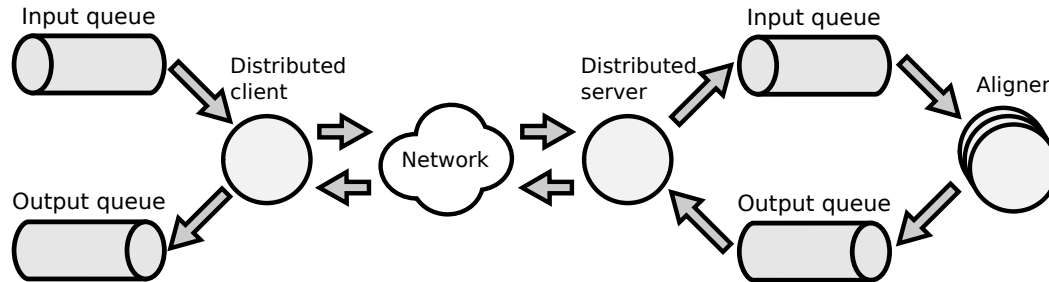


Fig. 7.6.: Distributed architecture, modified from author paper [WMK15]

We can see that two new components have been introduced. The **Distributed client** component is integrated into the system at the place of the **Aligner** component. It is responsible to send raw sequences to the **Distributed server** component, which runs on a remote machine, as well as receiving the aligned sequences from the remote aligner. No other component had to be modified to integrate those two new components, which is thanks to our stream based approach, where the different components communicate through queues.

The **Distributed server** component is responsible to recreate the environment in which the **Aligner** component is able to run. This is done by recreating the raw sequence input queue, as well as the aligned output queue.

Recreating the same environment on the remote computer as a local **Aligner** component would expect, opens up the possibilities for complex setups to distribute the workload. It is, for example, possible for a remote aligner to redistribute the alignment workload further to another remote aligner. This can be useful when dealing with a restricted access to an internal grid, where only one computer is accessible from the outside (front end).

The distributed alignment component was implemented three times using different technologies, with two of them being closely related. We used RMI, which is included in the Java standard library, DIRMI², which is an improved version of RMI (mainly supporting bidirectional connections to traverse NAT devices) and POP-Java (Chapter 8). All three technologies can be used alongside each other, in addition to local aligner threads. Nevertheless we use a single technology at a time to distribute the workload. It is also possible to completely offload the alignment task from the computer starting the alignment, which can be useful if for example a laptop computer starts the alignment process, but does not have the resources required to perform it locally.

The stream based approach used in this architecture has also other advantages. Every distributed Alignment component keeps track (and a copy) of all sequences that were sent to the remote machine, and only discards those copies once the alignment has been sent back. If a remote machine crashes or becomes otherwise unavailable, the local connection to the remote aligner can put the sequences to align back on the input queue, so that they

²<https://github.com/cojen/Dirmi>

can be aligned by other, still running aligner components. This allows for an easy way to be fault tolerant.

The other advantage the stream based approach has is load balancing. Every **Aligner** component will take the amount of sequences he is able to process, faster **Aligner** components will take more, slower less. By using queues to communicate, the load balancing happens naturally, as the workers get more work when they need it and not the other way around. The size of the queues is adapted dynamically, based on how full they are. This allows us to adapt to different computer speeds as well as network speeds.

When sending the raw sequences to the remote machines, we try to reduce the bandwidth usage of this transfer. This is done through mainly two approaches, efficient encoding, as well as only sending the required information. A raw read is composed of three components, its name, a DNA sequence as well as quality information associated with every sequence. The sequence name is not required for the aligner, so it is not transferred to the remote computer, the same goes with the quality information, which our aligner does not use. Both those information are later restored, when the alignment for the sequence is retrieved. The last part of the read, the actual DNA sequence to be aligned, is encoded using a one byte for every 3 nucleotides. This greatly reduces the amount of data to be transferred, making the distribution also viable over the internet.

This is especially useful when using the aligner in combination with a public cloud service. We implemented support for the Amazon AWS cloud service ³ directly in the aligner, allowing the user to specify the amount and types of virtual machines to be instantiated in the cloud to support the alignment process. Those virtual machines are dynamically created and destroyed at the beginning and end of the alignment process. The support for the Amazon AWS cloud was implemented through the use of the JClouds ⁴ library, a multi-cloud library developed by Apache. The support for this allows the users of the aligner which do not have the required infrastructure for alignment available locally, to benefit from the available cloud infrastructures.

But using cloud services comes at the price of privacy, something many laboratories cannot accept when handling human data. Even with cloud services having become more secure over the years, the moment the DNA is sent over the internet it has to be viewed as compromised. This is especially true when the cloud provider resides in another country. Our sequence aligner does not take any additional steps outside of anonymizing the data. But some approaches exist to lessen the privacy risks when sending human DNA data over the internet, such as mixing multiple samples during the alignment.

7.1.4. Results

The GensearchNGS aligner presented in this chapter has been used in hundreds if not thousands of real samples in various laboratories. In this section we analyse its performance in a more controlled environment, using simulated datasets for which the correct solution is known.

We evaluated our stream processing based aligner using multiple simulated datasets to determine both the quality of the alignments as well as the performance compared to three aligners. Namely, BWA-MEM (0.7.12-r1039), Bowtie 2 (2.1.0) and CUSHAW2 (2.4.3), which are three well established aligners in the research community.

³<https://aws.amazon.com/>

⁴<https://jclouds.apache.org/>

Aligner	Time	Aligned	Correct	Incorrect	Precision	Alignment rate
<i>BWA-MEM</i>	1188s	7880061	7780078	99983	98.73%	98.95%
<i>Bowtie 2</i>	1615s	7878771	7604671	274100	96.52%	98.94%
<i>Cushaw2</i>	799s	7868183	7650416	217767	97.23%	98.80%
<i>GensearchNGS</i>	1845s	7772837	7655461	117376	98.49%	97.61%

Tab. 7.1.: Test results for the *150bp-se-large-indel* dataset

We analyzed multiple datasets for our tests. The first datasets come from the genome comparison & analytic testing project (GCAT) [HWK⁺15], which is a project that provides sample datasets. Those datasets have known “solutions” and can be used to compare the performance of multiple aligners. The projects allows the user to download the datasets as raw data and upload the resulting alignment back to the website. The statistics of the amount of correctly and incorrectly aligned sequences is then provided through a convenient web interface.

We choose three datasets from GCAT to test the alignment quality as well as the performance of the aligners.

The first dataset is a single end dataset called *150bp-se-large-indel*, with 7’963’498 sequences of length 150 bp, that includes large indels. The two other datasets from GCAT are both paired end datasets. The first one being *150bp-pe-small-indel*, which has 3’981’750 paired end sequences of length 150 bp with small indels. The second one is *250bp-pe-large-indel*, which has 2’389’050 paired end sequences with a length of 250 bp and large indels.

The two main measures we make is the time required to align the dataset, as well as the precision and alignment rate of the aligner. We define the precision as: $P = \frac{c}{c+w}$ where P is the precision, c is the amount of correctly aligned reads and w is the amount of reads aligned incorrectly. The alignment rate is defined as: $A = \frac{c+w}{r}$, where r is the total amount of reads in the dataset.

All tests have been performed on a quad-core Intel(R) Xeon(R) CPU E5-1620 v3 @ 3.50GHz with Ubuntu 16.04 as the operating system and 16 GB of available RAM.

The tables 7.1, 7.2 and 7.3 contain the results for the described GCAT datasets.

The first Table 7.1 shows the results of the *150bp-se-large-indel* dataset. We can see the different aligners having very different performance results, with CUSHAW2 being the fastest. Our own aligner is the slowest, but not by much and has comparable performance to Bowtie 2, but still about twice as slow as CUSHAW2.

In regards to the quality of the alignment, BWA-MEM has the highest alignment rate and our aligner the lowest. Interestingly, our aligner does quite well in terms of the alignment precision, with very few incorrectly aligned sequences. With both precision and alignment rate high for this dataset, the BWA-MEM aligner can be considered the best.

The second dataset *150bp-pe-small-indel* looks at a similar dataset as the previous one, but this time using paired end sequences and smaller indels. Table 7.2 shows the result of those tests. Sadly, CUSHAW2 was not able to correctly finish this test, as the alignment file it produced did not maintain the correct names of the aligned sequences, making the analysis impossible. It is it to note however, that CUSHAW2 aligned the dataset fastest, in 827s. As for the performance of the 3 other aligners, they are more or less equivalent, with BWA-MEM having a slight advantage. The quality of the alignments is very interesting, as Bowtie 2 seems to have trouble aligning the data correctly (seen by a low precision rate).

Aligner	Time	Aligned	Correct	Incorrect	Precision	Alignment rate
<i>BWA-MEM</i>	986s	7879143	7837812	41331	99.48%	98.94%
<i>Bowtie 2</i>	1001s	7769798	7253123	516675	93.35%	97.57%
<i>GensearchNGS</i>	1002s	7832371	7728547	103824	98.67%	98.35%

Tab. 7.2.: Test results for the *150bp-pe-small-indel* dataset

Aligner	Time	Aligned	Correct	Incorrect	Precision	Alignment rate
<i>BWA-MEM</i>	1105s	4727769	4706284	21485	99.55%	98.95%
<i>Bowtie 2</i>	1158s	4660702	4505942	154760	96.68%	97.54%
<i>GensearchNGS</i>	1083s	4719481	4688331	31150	99.34%	98.77%

Tab. 7.3.: Test results for the *250bp-pe-large-indel* dataset

Our aligner and BWA-MEM have similar alignment and precision rates. The difference to the previous dataset can be explained by the usage of paired end sequences. When aligning paired end sequences, a sequence that is hard to align can be “saved” by its pair which may be more easily alignable. Depending on how an aligner exploits this fact, it will improve the quality of the alignment.

The last dataset 250bp-pe-large-indel shows similar results as the previous one. We can see that the longer sequence lengths affect the alignment speed of our aligner the least, but all three aligners remain similar in performance. The same pattern in terms of alignment quality can be seen again, with both BWA-MEM and our aligner being better than Bowtie 2.

We can see from the performed tests that our aligner gives similar results as the other aligners with comparable performance, at least in the case of paired end sequences.

Distributed alignment

To test the performance of the aligner in a distributed environment we decided to use a testcase which is typical for a small laboratory which only have limited computing power at its disposal. The results of this section are similar, but more detailed, to our publications [WMK15] on the subject.

In a small laboratory, it is common to find computers with different configurations as well as the need to include external computing resources to align large NGS datasets in a timely manner. To test this we decided to create 4 different configurations to test not only their performance, but most importantly the flexibility of the distributed alignment approach we implemented.

For the tests we used a Laptop (dual core Intel Core i7-3520M 3.6 Ghz, 8GB RAM) as the main computer to launch the alignment, in conjunction with a desktop computer (Quad core Intel Core I7 870 3.6 GHz, 8GB RAM) and Amazon AWS EC2 cloud instances (c3.xlarge, Quad core Intel Xeon E5-2680, 7.5 GB RAM). The local machines were connected with a 1 Gb/s switch, which in turn was connected with the internet through a 100 Mb/s up/down connection. We defined four possible configurations to run the alignment:

- Config 1: Laptop
- Config 2: Laptop and desktop computer

- Config 3: Laptop, desktop computer and one Amazon AWS EC2 instance
- Config 4: Laptop as client, two instances on Amazon AWS EC2

In the last configuration, the laptop does not actively participate in the alignment process (other than reading the raw data and saving the resulting alignment file). This configuration is useful if the client machine, like the laptop, is underpowered and does not have the required 5 GB of RAM available. In this case the user can still work with his normal machine, but offloads the complete alignment process to a different machine, or in this case the Amazon cloud.

The connection between the different computers can be seen in Figure 7.7.

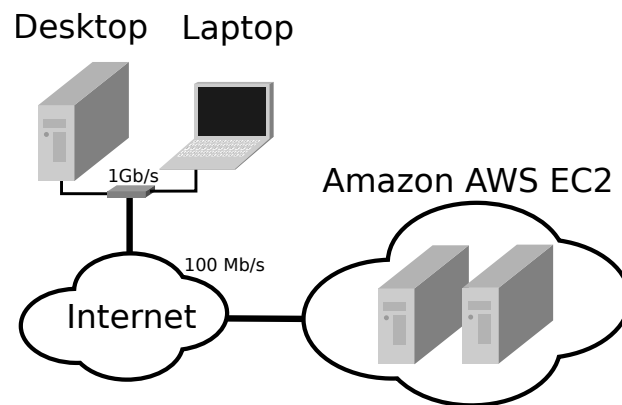


Fig. 7.7.: Benchmark setup

We also decided to test the configuration 4 in more detail. For this we repeated the tests with 1 to 10 Amazon AWS EC2 instances with the laptop as the client (but he does not perform alignment himself). This lets us better evaluate how well the distributed alignment scales over multiple computers.

For all configurations we only measure the time required for the actual alignment, excluding all the setup time required to start the different installations. This was mainly done because of the variable speed of the Amazon cloud, which can require between 5 to 20 minutes to launch an instance of the aligner and load the required data into memory. Also for every launch of the aligner, if an Amazon instance is used, it is started for that specific alignment. This makes the use of the cloud less interesting for small alignments, due to the setup times that become important.

The aligned dataset is from the GCAT project [HWK⁺15], called illumina-100bp-pe-exome-150x, consisting of 45'038'905 paired end reads of length 100 bp. To speed up the benchmarks, only the first 5 million reads were aligned against the human reference genome hg19. Table 7.8 shows the raw computation times, with Figure 7.9 presenting the same numbers graphically.

We can see that our aligner works well over the different tested configurations. We can also see, that for a small laboratory it can be interesting to use the cloud to replace one strong desktop machine, as the performance can be similar.

This raises the question of how well the alignment scales over multiple computers in the cloud, and up to how many can be used to efficiently align the data. The results of those tests can be seen in Table 7.10 and Figure 7.11.

The aligner does indeed scale well over 10 nodes, with near optimal speedup up to 5-6 nodes. This shows that our approach to how the distribution is handled is well adapted.

Configuration	Time
1 (Laptop)	1'465 s
2 (Laptop + Desktop)	427 s
3 (Conf. 2 + 1 AWS)	267 s
4 (2 AWS)	353 s

Fig. 7.8.: GensearchNGS raw distributed alignment times

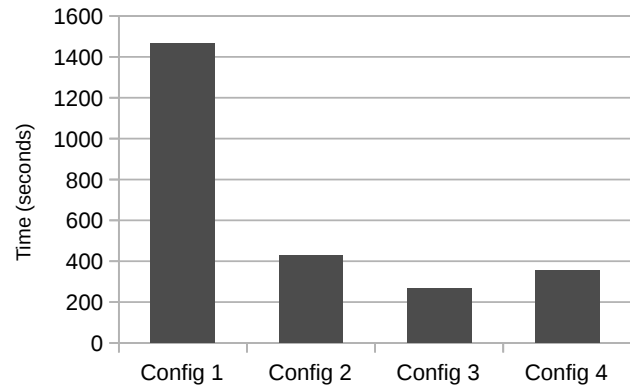


Fig. 7.9.: Alignment times

Configuration	Time	Speedup
AWS 1 instance	673 s	1
AWS 2 instances	353 s	1.9
AWS 3 instances	241 s	2.79
AWS 4 instances	181 s	3.71
AWS 5 instances	144 s	4.67
AWS 6 instances	120 s	5.6
AWS 7 instances	104 s	6.47
AWS 8 instances	98 s	6.86
AWS 9 instances	95 s	7.08
AWS 10 instances	94 s	7.15

Fig. 7.10.: Amazon AWS scaling over 10 nodes

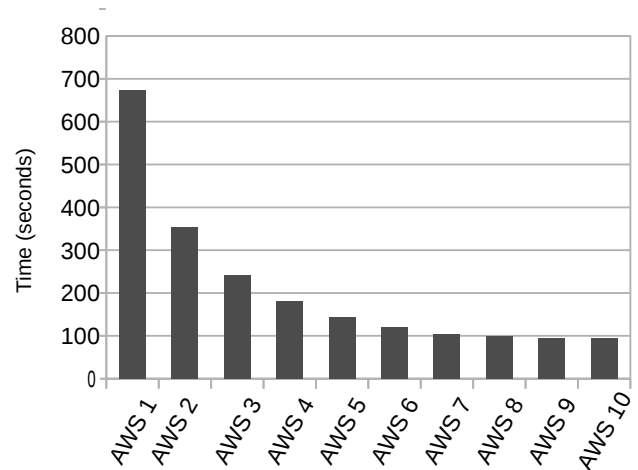


Fig. 7.11.: Amazon AWS EC2 alignment times over 10 nodes

We can also see that while the speedup remains interesting, after 6-7 nodes the benefit of adding more nodes is very small.

This shows that our aligner adapts to the different proposed configurations, showing its flexibility to adapt to various real life scenarios. The way the distributed alignment is approached, load balancing is done automatically and multiple machines can fail during the process. As long as the client machine (the one starting the alignment process), as well as at least one machine that does the alignment are still running, the alignment will finish correctly. This was tested by forcibly killing GensearchNGS on a remote machines and comparing the result with the results of a normal operation. The results were indeed equal in the various tests where the fault tolerance was tested.

7.1.5. Summary

In this chapter we discussed the creation of an aligner using the Java programming language, focusing mostly on paired-end sequencing data for humans. The goal was to develop a multi-platform aligner which can be used on platforms on which traditional aligners do not run (such as windows). While the goal was not to create a better aligner than the existing ones, but to create a solid base for further work, we could show that our aligner achieves similar performance and alignment quality as well established existing aligners.

We also showed how to distribute the workload of the aligner efficiently over a variety of resources. We could combine multiple local machines, as well as cloud resources, showing the ability to use the aligner in a variety of situations and to adapt to the needs of the individual users. As will be discussed later in Chapter 6, the aligner was integrated into the graphical NGS data analysis pipeline that we developed, to give the user the possibility to align NGS data on computers which do not have traditional aligners available.

7.2. Meta-Alignment

As discussed in the previous chapter 7.1, sequence alignment is a critical task when doing NGS data analysis. With the multitude of existing aligners today and the large variety of available options they have, it is very difficult to determine the best choice for a particular dataset. This is why we developed a new method to reduce the complexity of this task, by combining the output of multiple sequence aligners and choosing the best alignment for every sequence. This work has been presented at the European Human Genetics Conference 2016 *Meta-alignment: Combining multiple sequence aligners to improve alignment quality* as a poster presentation.

7.2.1. Introduction

Sequence alignment at its core is a simple process. For every sequence produced by the sequencer, the position on the reference sequence with the lowest *edit distance* is searched. As discussed in Section 3.1.1, the algorithms to calculate the edit distance are well known. In order to reduce the time required to test all positions on the reference sequence to find the one with the lowest edit distance, heuristics are used to reduce the search space. Different aligners and different options use different heuristics, leading to different results depending on the aligner. The differences between aligners are not only present between different datasets, but also inside the same datasets, where one aligner or setting works better for a certain part of the data than another. This makes sequence alignment and the choice of the right tools a non trivial task. This is why we propose a new approach to reduce this complexity. Our novel approach, called *meta-alignment*, combines the strengths of the different aligners to produce an unified alignment out of multiple separate alignments.

Sequence aligners output for every aligned sequence a location on the reference at which the sequence has been placed, as well as a so called CIGAR string. The CIGAR string indicates how exactly the sequence has been mapped against the reference at the indicated position, with most notably the information if and where indels have been detected. Aligners use a *score matrix* to determine what the best alignment for a sequence is. This score matrix indicates the score for a matching nucleotide, as well as the penalty for mismatches, indels and skipping parts of the sequence. Given a certain alignment of a sequence, the score can be calculated and thus compared between different aligners. This means that, given a certain score matrix, it is possible to rank the alignments given by different aligners to determine the most optimal alignment. It does not matter the specific method used by the different aligners, in the context of a (predefined or user specified) score matrix, the different alignments can be compared.

This is the core idea of our meta-alignment method. The output of multiple sequence alignments is taken and then the best alignment, based on a given score matrix, is stored in a new alignment file. The hypothesis is that using this approach, the quality of the alignment, as well as the number of aligned sequences can be improved.

In Section 7.2.2 we discuss implementation details as well as the test setup used to evaluate the validity of this approach. The results of those tests are presented in Section 7.2.3 with a discussion of the method in Section 7.2.4.

7.2.2. Method

To test the meta-alignment method, a stand-alone tool has been developed using the Java language based on the GensearchNGS pipeline (Chapter 6). In terms of external dependencies, the same libraries and technologies as in GensearchNGS are used. The tool does not use parallel or distributed computing, but focuses on the improvement of the alignment quality only.

Figure 7.12 displays an overview of the proposed method to combine the output of multiple aligners through meta-alignment.

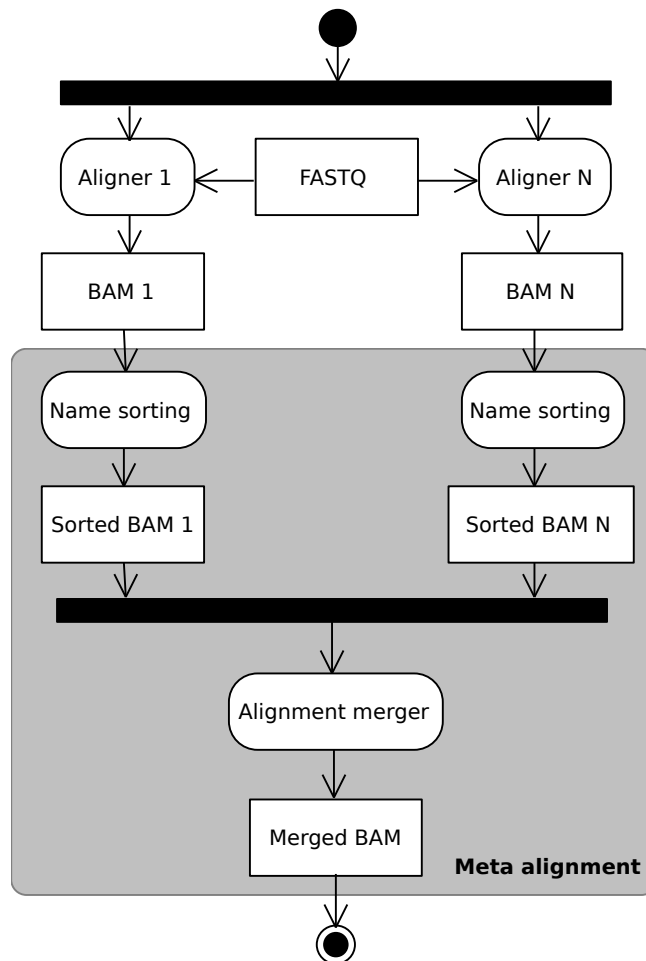


Fig. 7.12.: Activity diagram of the proposed meta-alignment method

As shown in the figure, the process starts by aligning the raw *FASTQ* data with multiple sequence aligners. The sequences in the alignment files are then sorted by their name, instead of sorting them by their reference position as is done usually. Those name sorted BAM files are used as the input for the **Alignment merger**, the heart of the meta-alignment method, creating a merged BAM files with the best alignment for every sequence. The following paragraphs look at this process in more detail.

The application is able to take multiple alignment files as an input and create a merged alignment file, using the best alignment for every sequence. The user is able to define multiple options on how the merging is done. Not only can he specify as many alignments (which have to originate from the same source file) as he wishes, but also the way that the

selection of the best alignment is done can be influenced. In the standard mode, for every sequence the best scoring alignment is used for the final alignment. The goal is to improve the overall quality of the alignment, and at the same time increase the total amount of aligned sequences. A sequence has only to be aligned by any of the input aligners to be used in the final alignment. The consequence is that although the total number of aligned sequences is expected to be higher than any individual aligner, the number of low quality alignments might also increase. This is because for certain reasons, a certain aligner might have rejected the alignment of a sequence for a good reason, where as another one still aligned it, even with its low quality.

This is why the user has the option to require a minimal number of aligners to agree on an alignment for it to be used. This is expected to decrease the amount of overall aligned sequences, but at the same time, to improve the quality of the alignment.

As specified, the input files are BAM files, usually coming directly out of the aligners. Those BAM files contain the alignments of all sequences. They usually are, by default, are either not sorted or sorted by position. To quickly find all the alignments of a particular sequence in the various alignment files provided by the user, the first step of meta-alignment is to sort the input files by sequence name. This step requires the sequences to have unique names and those names need to be the same in the different input files. If all alignments used as input come from the same source raw data file, this requirement usually holds true.

After the creation of the name sorted alignment files, all of them are opened in parallel and read sequence by sequence. As they are all sorted by name, finding all alignments for a specific sequence is very time effective.

Once all the alignments of a particular sequence have been recovered, the best scoring alignment needs to be determined. This is done using a score matrix, which for each nucleotide in the aligned sequence assigns a score based on its relation to the reference. Different approaches exist for score matrices, but in the context of meta-alignment we use score matrices with affine gap, just like the Gotoh [Got82] algorithm. This means, gap extensions have a different impact on the score than gap starts. This is because the addition of a new gap (either an insertion or a deletion) is less likely than the extension of an existing one.

To build the score matrix, we need the following scores:

- *Match* = α (When the sequence and the reference match)
- *Mismatch* = β (When the sequence and the reference mismatch)
- *Gap start* = γ (start of a deletion or an insertion)
- *Gap extension* = δ (start of a deletion or an insertion)
- *Skip* = ϵ (part of the read discarded by the aligner)

In our tests we used the following values for the score matrix: match = 2, mismatch = -3, gap start = -6, gap extension = -1, skip = -1.

To calculate the score of a specific sequence, we need to know how exactly it was mapped against the reference (and where). As mentioned earlier, every aligned sequence indicates its mapping against the reference through the use of a CIGAR string. This CIGAR string indicates where the sequence aligns to the reference and where it contains insertions and deletions. One important information is missing from the CIGAR string, which is if a

particular nucleotide matches the reference sequence or not. For this reason, every sequence needs to be mapped back (which means deletions have to be added and insertions cut out) to determine the exact score of every alignment. To determine this score, we need to determine the amount of matches (M), mismatches (MS), gaps starts (GS), gap extensions (GE) and skips (S) for that sequence. Once determine we can calculate the score as:

$$Score = \alpha * M + \beta * MS + \gamma * GS + \delta * GE + \epsilon * S \quad (7.2)$$

Once the score of every sequence is determined, the best scoring alignment is chosen and output to the final alignment file. If the user specified that multiple aligners have to agree on the position of a sequence, then after determining the best alignment, we count how many other alignments positioned the sequence at the same position.

To test the impact of meta-alignment on the results of sequence alignment we applied it to multiple datasets using three different aligners. The three aligners used were BWA-MEM 0.7.12-r1039 [LD09], Bowtie 2.2.6 [LS12a] and CUSHAW2 2.4.3. [LS12b]. Those are the same three aligners as were used to test our custom GensearchNGS aligner (Chapter 7.1). BWA-MEM and CUSHAW2 used the default alignment settings, Bowtie 2 was run with the option `-local`. We do not use our own aligner, as the goal of this chapter is to evaluate the meta-alignment approach and we considered it to be better to do so using well established aligners.

The first dataset used was a collection of simulated datasets, created specifically to study the effect of meta-alignment over various degrees of data quality. For this purpose we simulated 11 datasets with increasing degrees of errors in them. Each dataset consisted of 600'000 reads simulated using the human chromosome 19. The errors ranged from no errors to 20% errors with 50% of those errors being indels of length 1-3. The advantage of using simulated reads over real data is that the solution, which means the correct alignment for every sequence, is known. The results of this test are presented in Section 7.2.3.

The second dataset used comes from the genome comparison & analytic testing project (GCAT) [HWK⁺15] which provides standardizes test datasets. We used the *150bp-se-large-indel* dataset to compare the 3 aligners individually against the result of our meta-alignment approach. This is the same dataset which was used to test our aligner in Section 7.1. Section 7.2.3 discusses the results of this test.

For all datasets, we tested the effect of meta-alignment in 3 modes. The first mode is the default mode, which takes the best alignment for a sequence from all input alignments. The two other modes required at least 2, respectively 3, input alignments to agree on the position of a sequence for it to be used in the output alignment.

The next Section 7.2.3 presents the results of this tests and Section 7.2.4 gives a broader perspective on meta-alignment and its use-cases.

7.2.3. Results

This section presents the results obtained by applying our meta-alignment approach against the previously described datasets. The section is split in two parts, the first discussing the custom datasets with variable error rates and the second part using the datasets provided by GCAT.

Variable errors dataset

Using the first datasets, which consists of 11 datasets with gradually increasing error rates, we compared the precision and alignment rate of the three aligners as well as the meta-alignment approach used on the output of those same three aligners. All 11 datasets consist of 600'000 sequences originating from the human chromosome 19 (hg19). The error rates started with no errors and increased up to 20%. By default, those errors are SNPs (Single nucleotide polymorphism), but a certain percentage of them is instead created as indels of length 1-3 bases. The indel rate ranges from 0% to 50%, resulting in 10% indels (50% of the SNPs are converted into indels) in the worst dataset.

All 11 datasets have been aligned against the full human genome (hg19) by all three aligners. Afterwards the meta-alignment algorithm was applied to the output of all three aligners for every dataset, once with the default setting and also with a minimum of 2 respective 3 aligners needing to agree for the resulting alignment.

We calculated the precision as well as the alignment rate for all aligners and the meta-alignment approaches. The precision is defined as: $P = \frac{c}{c+w}$ where P is the precision, c is the amount of correctly aligned reads and w is the amount of reads aligned at the wrong place. The alignment rate is defined as: $a = \frac{c+w}{600'000}$ where a is the alignment rate.

Those two values give us an understanding of how much of the raw input data is used to create the resulting alignment, and to which degree the resulting alignment is correct.

We first look at the alignment rate of the different aligners and meta-alignment settings. Figure 7.13 we can see the alignment rate and how it evolves over various degrees of errors in the datasets.

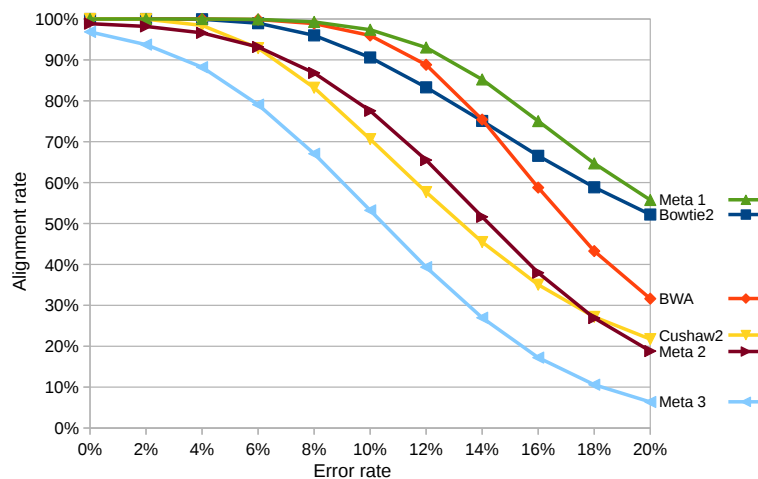


Fig. 7.13.: Alignment rate of the alignments ranging from no errors (Dataset 1) to 20% errors with 50% indels (Dataset 11)

We can immediately observe that the meta-alignment in its default mode (*Meta 1*) has consistently higher alignment rates than any other aligner. As the default setting of meta-alignment is to take the best alignment for every sequence out of all input alignments, this is the expected result. Any sequence only aligned by a single aligner will be found in the final alignment and thus the total count of aligned sequence will always be higher than any of the single aligners. Requiring two (*Meta 2*) or three (*Meta 3*) aligners to agree on the best alignment significantly reduces the amount of aligned sequences. This again is a

result that is expected, as in those two modes the goal is not to increase the total amount of aligned sequences, but improve the quality of the aligned sequences.

To analyze the quality of the aligned reads we look at the precision of the alignments. As already mentioned, the precision is defined by the percentage of correctly aligned sequences when only looking at the aligned sequences (not aligned sequences are not counted). Figure 7.14 shows the precision of the alignments over the same 11 datasets.

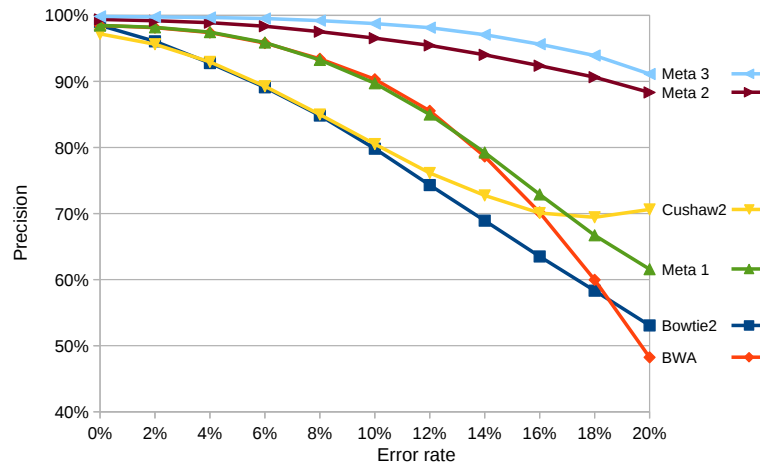


Fig. 7.14.: Precision of the alignments ranging from no errors (Dataset 1) to 20% errors with 50% indels (Dataset 11)

The meta-alignment approaches that require two or even three alignment to agree are consistently the ones with the highest precision. Even with the dataset which have the highest error rates, those two approaches reach 88.3% and 91.1%. In terms of precision, the default meta-alignment reaches 61.5% and is beaten by CUSHAW2 with 70.6%. But it has to be noted that the alignment rate of CUSHAW2 is only at 21.7%, which is very low compared to the other aligners.

What is interesting to see is that *Meta 2*, greatly improves the precision. The difference between requiring *Meta 2* and *Meta 3* is much lower, but comes with a big drop in the alignment rate as seen in the previous analysis.

Those measurements show the tradeoff between the amount of sequences aligned, and their quality. A hypothetical aligner that aligns every sequence at a random position would reach an amazing 100% alignment rate, no matter the quality of the input data. But the precision would be close to 0%. The output of meta-alignment cannot be better than the best aligner for the individual sequences, which limits the precision and alignment rates that can be reached. If for example no aligner aligns a certain sequence correctly, our method cannot correct this. With that in mind, the results of meta-alignment seem interesting, especially when considering the datasets with high error rates.

GCAT alignment

The genome comparison & analytic testing project (GCAT) [HWK⁺15] is a project which provides free datasets and a website to test different aligners. To test the accuracy of alignments, the GCAT project generated a set of datasets for which the correct alignment is known for every sequence. To test an aligner, the website provides the raw FASTQ files of every dataset. The raw data can then be aligned locally against the human reference

genome HG19 with any aligner, as long as a BAM file is generated. This BAM file can then be uploaded to the website which will automatically create the statistics of the correctly and wrongly aligned reads. We decided to test the meta-alignment method on one of the provided datasets, which is called *150bp-se-large-indel*. This dataset contains single ended sequences with a length of 150bp and a large amount of indels, even though the GCAT project does not specify how high the amount of added indels is. We used the same testing procedure as for our simulated data with the same aligners (BWA-MEM, Bowtie 2 and CUSHAW2) and also tested our meta-alignment approach with 1, 2 or 3 alignments that had to agree for an alignment to be used. Table 7.4 contains the details of the performed tests.

	Total	Correct	Wrong	Not aligned	Precision	Alignment rate
BWA-MEM	7'878'949	7'779'572	99'477	84'549	98.74%	97.69%
Bowtie 2	7'878'771	7'604'671	274'064	84'727	96.52%	95.49%
CUSHAW2	7'868'183	7'650'416	217'767	95'315	97.23%	96.07%
Meta 1	7'878'987	7'781'337	97'614	84'511	98.76%	97.71%
Meta 2	7'787'802	7'737'157	50'645	175'696	99.35%	97.16%
Meta 3	7'507'014	7'487'040	19'974	456'484	99.73%	94.02%

Tab. 7.4.: Meta-alignment comparison table for the *150bp-se-large-indel* dataset

We can again see the BWA aligner getting very good results, having the highest precision and alignment rate when used alone. Similar to our previous test, we can see how the meta-alignment is able to improve the quality of the alignment. The first meta-alignment configuration, Meta 1, is able to improve both the precision and the alignment rate compared to all other aligners. But the improvement, especially in terms of alignment rate, is rather minor, which indicates that all three source aligners had trouble with a similar set of sequences. Meta 2 and Meta 3 on the other hand greatly increase the precision of the alignments compared to single aligner. Especially the Meta 2 configuration, which requires 2 out of the 3 single aligners to agree, shows a great compromise between a decreased alignment rate and the improved precision.

When looking at the time required to perform the meta-alignment on this dataset, we get the following values using a quad core Intel(R) Xeon(R) CPU E5-1620 v3 @ 3.50GHz: BWA-MEM, 13 minutes 13 seconds. Bowtie 2, 29 minutes, 54 seconds. CUSHAW2, 15 minutes, 25 seconds. The times for all 3 meta-alignment approaches are the same, 32 minutes 53 seconds.

We can observe that the time required to perform meta-alignment is significant compared to the benefits of the method. As no focus was put on optimizing the method, there is still a lot of room for improvements in that regard. The way the algorithm works, it lends itself ideally to be distributed. This is a possibility we explored through a student project [BKW16] by Christophe Blanquet. The explored idea was to use POP-Java (Chapter 8) to distribute the different aligners used in the meta-alignment process, to execute them in parallel on multiple machines. The project was able to create an initial prototype which validated the idea and will be explored more in the future.

7.2.4. Summary

As shown through our tests, the meta-alignment approach shows great potential in certain use-cases, especially when working with high error rates in the data to align. The current

prototype has been published as free software as part of the GNATY project suite on <http://gnaty.phenosystems.com>. While several limitations still need to be addressed, like the lack of paired end support, the feedback from the community (which was gathered during the conference of the European society of human genetics (ESHG) 2016), was very positive.

Other than the lack of paired end support, the performance of the approach is a major obstacle. But as described, the algorithm used lends itself perfectly to be parallelized and distributed, an approach that has already been explored through a student project.

7.3. Variant calling

Variant calling (see Section 3.1.1) is a central aspect of NGS data analysis when doing DNaseq and diagnostics. Having a precise and fast method to call variants in a sample is a key aspect of efficient data analysis. In this section we present GNATY, a variant caller developed with speed in mind. GNATY was first presented in a poster *GNATY: A tools library for faster variant calling and coverage analysis* [WKD15] at the GCB 2015. It was later fully released and presented during a conference paper *GNATY: Optimized NGS variant calling and coverage analysis* [WKD16a] at the IWBBIO 2016. This chapter is based on the content of those two works and thus also discusses the subject of coverage analysis, which is used as a second use-case to validate the proposed architecture for high performance variant calling.

7.3.1. Introduction

As discussed earlier (See Chapter 2), the amount of data NGS technologies produce is growing increasingly fast. Being able to analyze them in a timely fashion, without requiring a huge infrastructure is key for the adoption of this technology. Analysing NGS data has many elements: Filtering the raw data, aligning it against a reference, calling variants etc. Many of those analysis steps received a lot of attention in recent years to optimize them. In particular sequence aligners received many improvements, resulting in a multitude of competing sequence aligners. But not all aspects of NGS data analysis received the same focus, and we believe that there is still room for improvement in those areas. This is why in this chapter we are looking at variant calling, as well as coverage analysis as targets for speed improvements.

The goal is to improve the performance of those domains, without impacting the quality of the analysis. The resulting tools have been released as a free tool called GNATY (short for GensearchNGS Analysis Tools librarY)⁵. GensearchNGS is the graphical pipeline developed during this thesis and discussed in Chapter 6. The way we try to approach this goal is to use modern development techniques, such as stream processing and multithreading, to optimally use the resources available on a computer. Our goal is to use existing computing infrastructure to analyze more data as well as to use infrastructure currently not suitable (such as laptops) to perform this type of analysis.

The target of our optimization is variant calling, which is performed on an alignment file, usually in the BAM file format [LHW⁺09]. This file contains all the aligned sequence, separated into groups based on the reference (usually a chromosome) on which they have been aligned. The goal of variant calling is to find the positions in the alignment which differ from the used reference sequence. Those differences, which can come in the form of SNPs or indels, are called variants. A variant caller, based on its settings, will produce a list of variants described by their position on the reference and the difference observed between the reference and the alignment at that position. We describe a generic stream based architecture to speed up this type of analysis. To show that our approach is valid also for other types of analyses, we also apply it to coverage analysis, which is a similar problem in many ways.

The rest of the chapter is structured as follows. We look at the different existing variant callers and methods in Section 7.3.2. Section 7.3.3 discusses the implementation and meth-

⁵<http://gnaty.phenosystems.com>

ods used to create our more optimized variant calling tool. In Section 7.3.4 the performance of the tool is analyzed followed by a discussion of this chapter in Section 7.3.5.

7.3.2. State of the art

Like for most of the NGS data analysis steps, many tools exist to perform variant calling. While they all have the same goal, namely detecting the differences between the aligned sequences of a sample and the reference, they approach this problem with different strategies. Two main groups of tools can be identified, the ones using probabilistic methods, mainly based on the Bayes theorem [BP63], and the ones using heuristic and statistical approaches. GATK [DBP⁺11] and samtools [LHW⁺09] are two popular variant calling tools, both members of the first group of tools using probabilistic methods. In the second group, using heuristic and statistical methods, the main tool used is Varscan 2 [KZL⁺12], which is also the tool we based our approach on, as it is the preferred tool of the human geneticists laboratory at the University of Würzburg. It has also been shown recently [WANW14] that both approaches create similar results when using the correct parameters. The cited paper compares Varscan 2 with other approaches, in particular GATK. As we try to reproduce the exact analysis results of Varscan 2, the comparisons of the cited paper between Varscan 2 and the other variant callers also applies for GNATY.

The generic architecture we propose not only applies to variant calling but for other types of NGS data analyses. For this purpose we also look briefly at coverage analysis of an alignment file, a process in many ways similar to variant calling. For this work, we consider coverage analysis to be the creation of an annotation file in the BED format, containing the coverage information for every position of the alignment. A couple of tools exist for this task, but the best known one is BEDtools [QH10]. BEDtools is a collection of tools which either work with BED files or produce BED files. Those BED files can be used for other data analysis tools as well as for visualization in genome browsers. The coverage analysis we implemented is based on the method used in BEDtools.

7.3.3. Methods

Variant calling is a mainly I/O (Input/Output) bound analysis. Determining if at a certain position in the alignment there is a variant is not costly in terms of calculations, but the files which contain this information can be very large and become a bottleneck. This is why when designing the architecture of GNATY, we focused on splitting the data analysis in various modules. They can work independently and most importantly separate the calculation heavy and I/O heavy operations. This way, during the time data is loaded from storage (or written), the processor can continue to analyze the already loaded data, without having to wait for the I/O operation to finish. This overlap between communications and computation is an important aspect when writing high performance applications.

This is why we decided to use a stream based approach, where I/O and calculation tasks run in parallel. The application has been developed as a Java application using Java 7 or newer.

Figures 7.15 and 7.16 show the proposed algorithm, once for the variant caller and once for the coverage analysis.

Thanks to the modular streaming approach, a lot of code can be shared between the two analysis tools. Indeed, the two modules which amount for most of the code complexity are

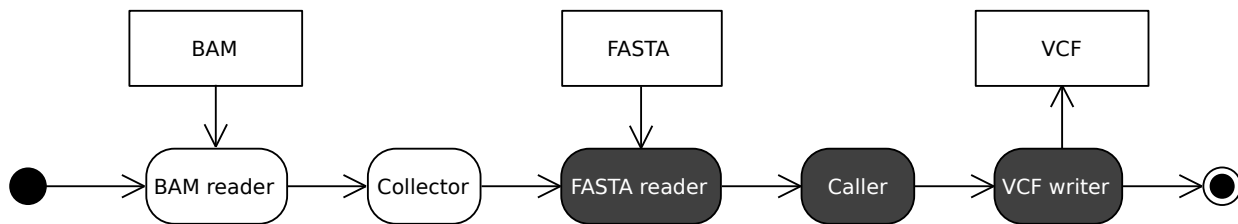


Fig. 7.15.: UML (Unified modeling language) activity diagram for the variant calling part, based on the author paper [WKD16a]

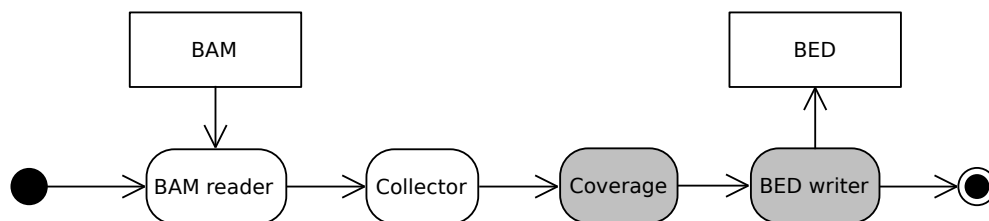


Fig. 7.16.: UML (Unified modeling language) activity diagram for the coverage analysis part, based on the author paper [WKD16a]

shared, the `BAM reader` and the `Collector` modules. The `BAM reader` is responsible to read the alignment file and outputs a stream of aligned sequences. To read the alignment files the Java library `Htsjdk`⁶ is used, which is part of the `samtools` project [LHW⁺09].

After the `BAM reader`, the `Collector` receives a stream of sorted sequences. Based on the sequences, the `Collector` constructs a data structure which contains the coverage information for the current region of the alignment. This information is stored in a circular buffer, which is expanded when needed. The length of this circular buffer is the length of the longest sequence in the dataset. As the incoming reads are sorted by their start position, every time a new sequence arrives the region before the start of that sequence can be analyzed (as no new information about it will arrive). At this moment, the information of the completed region is sent to the next module, which depending on the type of analysis is either the `Coverage` or `FASTA reader` module. For coverage analysis, the rest of the process is simple. Based on the information received from the `Collector`, the `Coverage` module calculates the coverage for every position. This coverage information, is sent to the `BED writer` which saves the information into a `BED` file.

For variant calling, the `FASTA reader` intercepts the stream of information coming out of the `Collector`. It is at this point that the reference information is added to the alignment information produced by the `Collector`. To call variants, unlike coverage analysis, it is necessary to not only know the aligned sequences at a certain position, but also the reference (as we search for differences between the alignment and the reference). This task is done by the `FASTA reader` which annotates the stream data with the reference. This means it can read only the required parts of the reference `FASTA` file, and that in sequential order (which at least for traditional hard disk is a speed advantage). This annotated alignment information is then streamed to the `Caller` which does the actual variant calling.

The variant calling is done using various user configurable options, such as minimum frequency, minimum coverage or minimal p-value. The p-value (probability-value) determines the probability that a variant is an artifact. It is calculated the same way as `Varscan 2`

⁶<http://samtools.github.io/htsjdk/>

does it, with Fishers exact test and a prior of 0.02. To calculate the Fishers exact test to determine the probability that an observed variant is different from a random sequencing error we need: The coverage x of the position at which the variant has been observed. The amount or reads r that support the variant. And the prior P which stands for the probability of having a sequencing error. We use Fishers exact test with the values $a = x * P$, $b = x - a$, $c = x$ and $d = x - r$. Thus, using Fishers exact test formula 7.3:

$$p = \frac{\binom{a+b}{a} \binom{c+d}{c}}{\binom{a+b+c+d}{a+c}} \quad (7.3)$$

We get the following formula 7.4 to calculate the probability:

$$p = \frac{\binom{x*P+x-x*P}{x*P} \binom{x+x-r}{x}}{\binom{x*P+x-x*P+x+x-r}{x*P+x}} \quad (7.4)$$

Only r and x are needed to calculate the probability, P is given by the algorithm (0.02 by default). For high coverage regions, the computational complexity of this method is too heavy and thus replaced by the Chi-square test. The Chi-square test gives very similar results at higher coverage values, but is much less computationally expensive. We use Fishers exact test when $a + b + c + d < 200$, and above that value the Chi-square test. This allows us to determine variants accurately even at high coverage locations in an alignment, where as other variant callers down-sample the alignment information for high coverage regions.

The result of the variant calling module is then sent to the `VCF writer` which outputs a VCF file of the variants found.

Worth mentioning are some special options integrated into our variant calling algorithm in relation to the reference Varscan 2 implementation. In contrast to the coverage analysis which reproduces the BEDtools output, the variant caller required more effort to reproduce the exact same results.

Varscan 2 has a few particularities when calling variants, if not to say bugs, which had to be reproduced to get the same results and thus a fair comparison. Those adaptations which replicate the exact results from Varscan 2 have been put behind two special configuration parameters the user can activate. The first option, `-one`, is used to only call one variant at a certain position. Varscan 2 will always only report variant at a position. This can cause problems when a sample contains two heterozygous variants at the same position. Only the one with a higher frequency will be reported. We reimplemented this optional behavior to match the Varscan 2 method.

The second option, `-var2` is a broader collection of small changes that had to be made to reproduce the exact Varscan 2 behavior. The most important one being the way indels are handled. When calculating the coverage for deletions, Varscan 2 uses the coverage of the position just before the deletion, not like one would expect the coverage at the first position of the deletion. Figure 7.17 shows this difference in an example.

Varscan 2 considers the position marked in blue for the coverage of the one base deletion. GNATY on the other hand by default considers the red position for the coverage of the deletion. This can lead to differences between the two tools, like in this example if there is a minimal coverage of 3 for variants to be called.

Also, in certain situations, Varscan 2 calls variants at positions at which the reference is N, a letter that is used as a wildcard for all other nucleotides. This should not happen, as the nucleotide N is a placeholder for every possible nucleotide, thus making it impossible to find

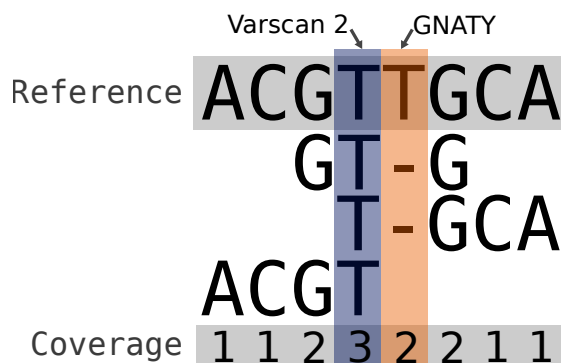


Fig. 7.17.: Difference between GNATY and Varscan 2 on default settings for indel coverage

any SNPs at those positions. We reimplemented this behavior and the user can activate it using the optional argument `-var2`, an option we also used during our tests. Those findings have been reported to the developers of Varscan 2, but as of today no change has been made in Varscan 2 to address those issues. For this reason those 2 options are available to the user in case they want to migrate from Varscan 2 to GNATY, without having to reevaluate the whole pipeline.

In the next chapter we look at how our implementation performs for both variant calling and coverage analysis.

7.3.4. Results

To test the performance of the variant calling as well as the coverage analysis, we used two datasets. The source of both datasets is the genome comparison & analytic testing project (GCAT) [HWK⁺15], a project which aims to provide standardized datasets for testing and comparing of different methods. The GCAT project provides a website with datasets as well as the ability to compare the results of different types of data analysis on those datasets between multiple methods. For the purpose of our tests, in which we compare the variant calling tool with Varscan 2 as well as the coverage analysis tool with BEDtools, the same datasets have been used for both use-cases. We picked two rather typical datasets originating from two different sequencing technologies. The first dataset comes from an ion torrent sequencer, called *ion-torrent-215bp-se-exome-123x*, which is a single end exome dataset with a coverage of 123x. It consists of 39'624'388 sequences of length 215 bp distributed over all chromosomes. The second dataset consists of 44'903'506 paired end sequences coming from an illumina sequencer. The individual reads have a length of 100 bp and the dataset is called *illumina-100bp-pe-exome-150x*. This dataset was also used to test the distributed sequence alignment in Section 7.1.4.

The alignment time for both datasets took 64 minutes (23 minutes for dataset 1, 41 minutes for dataset 2). To perform variant calling as well as coverage analysis, sorted BAM files are required. The unsorted SAM files produced by the aligner have been converted to sorted (by coordinates) BAM files with the use of samtools 1.0. This conversion process took 100 minutes in total, 37 minutes for the first dataset and 63 minutes dataset two. This produced the BAM files used for the benchmarks, with a size of 5.2 GB (dataset 1) and 6.8 GB (dataset 2).

The benchmarks have been performed on a computer equipped with two quad core CPUs of the type Intel Xeon E5-2609 clocked at 2.5 GHz. The computer was equipped with 32

GB RAM and a hard disk with a read speed of 128 MB/s and a write speed of 94 MB/s.

Variant calling

The results of variant calling is the time required to perform variant calling with both GNATY and Varscan 2 on both datasets. Varscan 2 as well as GNATY use the variant calling options recommended by [WANW14]. Those options are a minimum of 20% frequency, a minimum coverage of 10 and at least 4 supporting reads for a variant. In addition, Varscan 2 used the `-B` option which disables the probabilistic realignment feature (Varscan 2 recommends disabling this feature). Also the `-x` option has been enabled, which disables the feature to reduce the quality of overlapping read pairs, a feature not yet supported by GNATY. To create comparable results, GNATY uses the option `-one` (which only outputs one variant per location) and `-vs2` (which enables a mode that reproduces the way Varscan 2 calls variants). These options are used to guarantee that both tools output a list of variants as similar as possible, allowing for a fair comparison between them. Varscan 2 uses as its input the output of samtools, which comes in a special format called mpileup. The output of samtools has been sent directly to Varscan 2 using a standard Unix pipe, to avoid unnecessary hard disk access. The output of both tools is a VCF file containing the variants present in the datasets.

The tests have been repeated 3 times for every dataset. The output files of those tests have been compared to assure the same results of the analysis. Indeed both tools produced the same set of variants, with the exception of one variant. Both tools called 425'866 variants over both datasets, with one variant being the exception, as it was only called by GNATY. In practice, that particular variant does not make a difference in the data analysis. The particular variant in question is a SNP at a position where the reference is N, which means any nucleotide is accepted. GNATY only calls variants at those positions when in the Varscan 2 compatibility mode, to reproduce a bug in Varscan 2. Any downstream variant analysis tool will ignore variants at those positions, which means this difference does not change the results of the data analysis.

Figure 7.18 shows the variant calling times for both tools on both datasets.

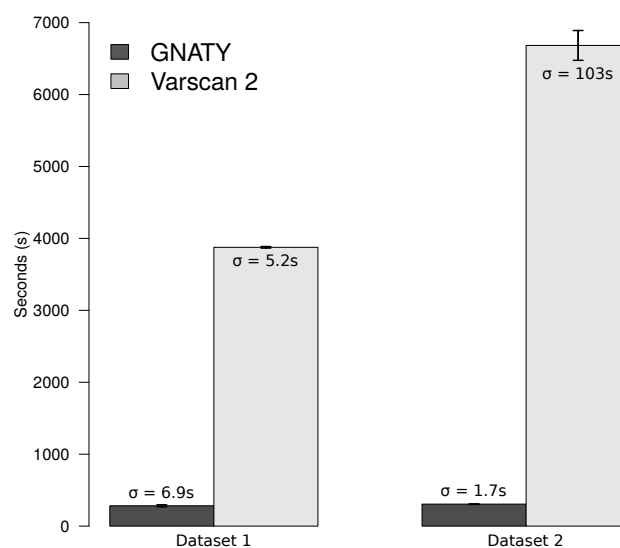


Fig. 7.18.: Comparison of variant calling times for GNATY and Varscan 2 on the two tested datasets. Source [WKD16a]

We can observe a massively faster analysis time for GNATY than for Varscan 2. Both datasets combined were analyzed in 9 minutes 49 seconds (589s) with GNATY, with 283s (σ 6.9s) for the first dataset and 306s (σ 1.7s) for the second dataset. Varscan 2 needed much longer, in fact 2 hours and 55 minutes (10'559s) to analyze those two dataset. This amounts to 1 hour 4 minutes (3'876s, σ 5.2s) for dataset 1 and 1 hour 51 minutes (6'683s σ 103s) for the second dataset. This gives an overall speedup of 18 times for our algorithm implemented in GNATY, for the same analysis results.

To put this numbers into context, Varscan 2 took about the same time it took to align the raw data and convert it into a sorted BAM file. GNATY on the other hand only required about 6% of the time to create the BAM file to analyse, which makes it highly interesting as a replacement tool for Varscan 2.

We also evaluated the impact the stream based approach has on our algorithm, to determine how much of the speed gain comes from the improved sequential code, and how much by running the different components in parallel. We did this by approximating a sequential version of the algorithm by reducing the size of the queues that connect the different modules. By reducing the size of all queues to 1, we approximate the performance of a sequential version. With this version, the Dataset 1 was analysed in 380 s, which is 34% slower than the normal version. For Dataset 2, the results were similar, with a 49% slower execution, bringing the time to 457s. Both of those version were still significantly faster than Varscan 2, which shows that a big part of the gain comes from a more efficient sequential algorithm.

Coverage analysis

The same two datasets were tested with the coverage analysis of GNATY, to show the flexibility and validity of the proposed architecture to other types of analyses than variant calling. GNATY is compared to BEDtools 2, reproducing its exact results. The results of the coverage analysis, which for both tools is a BED file, is identical, which makes it possible to replace one tool by the other without having to modify an existing analysis pipeline. The linux tool *md5sum* was used to calculate the MD5 hashes of the result files, and they are indeed equal between both tools. Both tools were ran 3 times on every dataset, just like for variant calling.

Figure 7.3.5 shows the overview of the benchmark results of both tools.

GNATY required 599s (with a σ 0.6s) to perform the coverage analysis for both datasets. For the same datasets, BEDtools 2 required 1260s (σ 12s), which is about twice the time that GNATY required to produce the same files. We can observe that the times required with our algorithm per dataset (291s for dataset 1 and 309s for dataset 2) are very similar to the time required to perform variant calling on the same datasets. This shows that we are mostly I/O limited, and not computationally limited. Calculating the coverage of an alignment file is much less complicated than calculating the variants present in the alignment. What really changes in the data analysis is the amount of data that needs to be written to the disk for the result files. For variant calling, the resulting VCF amount to about 32MB. But for coverage analysis, the BED files are 2.6 GB big, which results in a much bigger strain for the hard disk.

Figure 7.20 shows the total time required to go from the raw sequencing data of both datasets to the variant list and coverage analysis.

We can observe that using our tool, the overall time of the analysis is reduced by almost 50%, going from 360 minutes down to 183 minutes. This is mostly due to the faster variant calling, and in part because of the faster coverage analysis. Both the alignment phase and

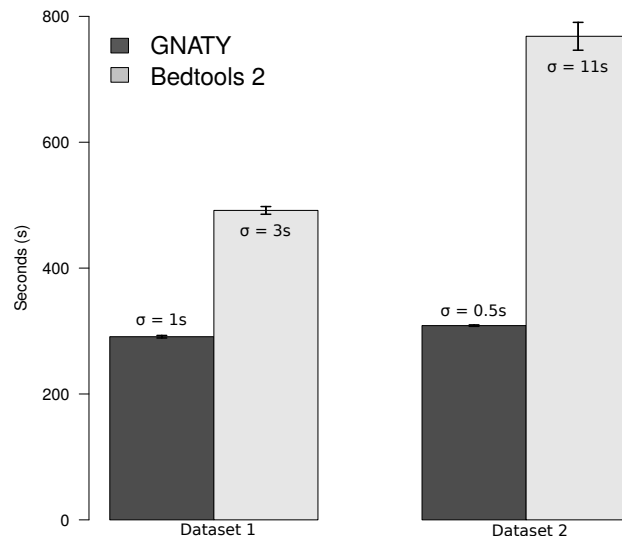


Fig. 7.19.: Comparison of coverage analysis times for GNATY and Bedtools 2 on the two tested datasets. Source [WKD16a]

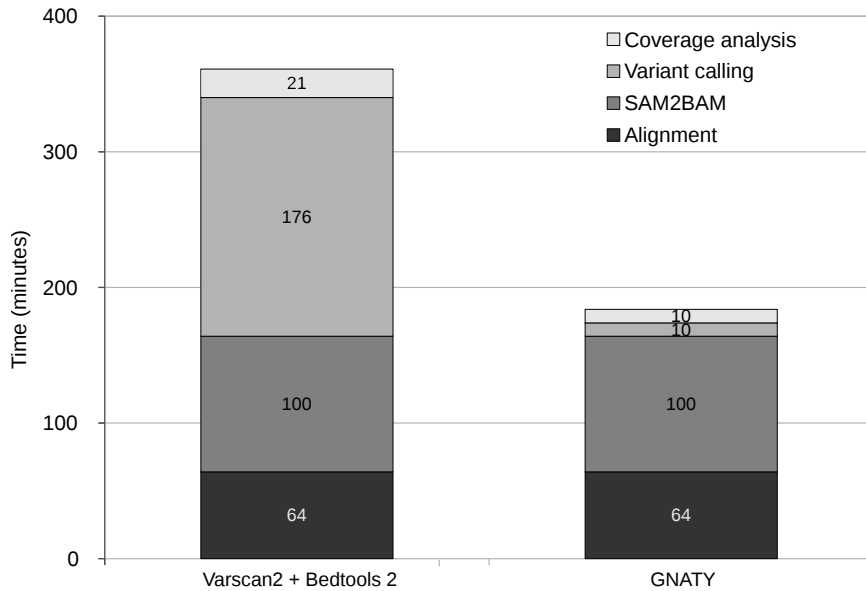


Fig. 7.20.: Total time required to go from raw sequencing data to a variant list and coverage analysis

the conversion from unsorted alignment files sorted alignment files where not optimized in this work.

7.3.5. Summary

We demonstrated that even well-established tools in today's NGS data analysis pipelines have still room for improvement without sacrificing the quality of the analysis results. We did this by implementing more efficient tools which replicate the results of existing tools. The speed improvements we achieved came both from more efficient sequential code, as well as the use of a modern software architecture which fully exploits the parallelism of modern computers. The proposed stream based architecture has been tested on two types of analyses, variant calling and coverage analysis. We could show that the separation of I/O tasks from computational tasks is able to give massive speedups in terms of analysis speeds. Variant calling with our implementation, while giving the same analysis results, resulted in a speedup of about 18 times over 2 datasets compared to a standard tool, Varscan 2. A similar but less pronounced effect can be seen when doing coverage analysis, which has a speedup of 50%. For coverage analysis, the algorithm was compared to BEDtools 2, and produces a binary equal results file, making it easy to replace BEDtools 2 in an existing pipeline. This shows that for some NGS data analysis steps, there is still a big potential for speed increases even without inventing new methods that change the analysis results. This is important in the context of the ever increasing amounts of NGS data, where many solutions are searched by increasing computing power or expanding to grids or clouds to cope with new data. Our work helps to make NGS data analysis more manageable on existing hardware, and also make it possible to perform the analysis on smaller computers, such as laptops.

We intend to apply our approach to other variant calling methods, especially probabilistic variant calling like GATK and samtools use it, is a priority. Preliminary measures, even if not exactly comparable as the results differ between both methods, show a speedup of 5 when using GNATY compared to GATK or samtools for variant calling. While this speedup might change when implementing a probabilistic variant calling method, it shows an important potential for improvements.

We also would like to explore the possibility to apply our proposed architecture on other types of analyses. We are also interested in improving the underlying libraries used, especially HTSjdk, which is used to read and write BAM files. While we already provided some patches to improve the performance of HTSjdk for some of our use-cases, more could be done, especially for faster decompression of BAM files, which is a major bottleneck when reading alignment files.

7.4. RNA-seq

This section describes the work that has been done to integrate RNA-seq data analysis into the graphical pipeline presented in Chapter 6. The work presented here also helped to publish a journal publication, *Non-Coding RNAs in Lung Cancer: Contribution of Bioinformatics Analysis to the Development of Non-Invasive Diagnostic Tools* [KWS⁺17] with the author as co-author.

7.4.1. Introduction

Understanding which genes are expressed inside a cell as well as the exact RNAs they produce is important to understand the biological functions going on in that particular cell and in the sample in general. While DNA-seq can give a general understanding of what might happen in the cell given its DNA, looking at its RNA gives us a much better understanding of the actual processes happening. More about how RNA works and its role inside our cells can be found in Section 2.1.1 and Section 3.2.

There are many ways to look at RNA-seq data and to answer biological questions by analyzing it. What we want to focus on in our work is the differential expression of genes between multiple samples. While detecting novel transcripts and genes is a highly interesting research question, it does move away from our main goal to lower OMICs data analysis complexity, targeting diagnostics.

To detect differentially expressed genes, we can use a large portion of the general DNA-seq pipeline described in Section 3.1 and Chapter 6. We can reuse the initial quality control step of the DNaseq pipeline, as RNAseq and DNaseq both use NGS data using the file formats.

At the sequence alignment step there are differences between both analyses. As the RNA-seq data does not originate from all over the genome (like DNA-seq data), but only from the transcript regions, the mapping process is slightly different. As discussed in Section 3.3.1, there are two approaches when aligning RNA-seq data against a reference. Either a full genome reference can be used, or a specialized transcriptome reference is used, which only contains the references of the RNA coding proteins in the genome. Both approaches have their merits, but to be more consistent with the rest of the supported data analysis, we focused on using a full genome reference, without creating any obstacles for the use of a transcriptome reference. To align RNA-seq data against a genome reference, specialized aligners are commonly used. Those specialized aligners take into account the fact that reads are not mapped continuously to the reference, but that large gaps exist where the introns of the transcripts can be found. This type of alignment is called spliced alignment.

Once the data is mapped, in many regards the same types of analyses can be performed with RNA-seq data as with DNA-seq data. For example variants can be called and compared to a DNA-seq dataset of the same sample. The alignments can as well be visualized in a genome viewer, which is why we adapted our genome viewer to support RNA-seq data (more about this in Section 7.6.4). In this thesis we limit ourselves to the visualization of RNAseq data through a genome browser. Other visualization approaches, mainly regarding the interactome, have been explored during a project [Sis16], but did not yet lead to significant results.

7.4.2. Methods

In this section we look at the different features that were implemented to support RNA-seq data analysis in the graphical pipeline developed for this thesis. As with the rest of the pipeline, the Java programming language was used. We focus on the handling of the alignment of the data in Section 7.4.2. This is followed by a per sample gene expression analysis in Section 7.4.2. The individual samples are then compared in terms of gene expression to detect differentially expressed genes, as described in Section 7.4.2. Once the differentially expressed genes are determined, we have implemented a set of tools to analyze their features to better understand their relationship with the biological process happening in the cell (Section 7.4.2).

Spliced alignment

RNA-seq data commonly uses spliced alignment when aligning against a full genome. While many researchers use aligners like Bowtie [LTPS09], more specialized aligners such as TopHat 2 [KPT⁺13] and STAR [DDS⁺13] exist. Even though we developed a custom aligner for DNA-seq (see Section 7.1), this aligner was only experimentally extended with spliced alignment capabilities.

As the complexity of the task to create a full-fledged spliced aligner was out of the scope of this thesis, we decided to integrate aligners that support this type of alignment into the graphical pipeline we developed. While Bowtie was already integrated for the purpose of DNA-seq and could be used without modification for RNA-seq data analysis, we also integrated TopHat 2 into the graphical pipeline. This allows the user to use both of those aligners transparently without having to learn their usage on the command line and having to worry about handling the data directly.

While we don't directly support other aligners through the user-interface, we integrated an automatic detection for the STAR aligner when importing aligned datasets. This allows the user to transparently import externally aligned RNA-seq datasets, setting their datatype automatically to RNA-seq, which is important as some features are only displayed when looking at certain datatypes.

Gene expression

To implement gene expression analysis we based our method on the htseq tool [APH15]. Using our gene model, described in Section 6.6, we determine all genes, transcripts and exons that code for RNA in the genome that is being analysed. Going through all aligned reads, the sequences of every one of those regions are counted. For a sequence to be counted for a particular region, it has to overlap that region. For transcripts, we additionally determine if the sequence is likely to come from a different transcript from the same gene. To determine this, the start and end position of the sequence are looked at to determine if they map to a different transcript. In the case of paired end data, we also look at the mate pair position to determine if it maps to a different transcript. A sequence can map to multiple regions.

The data is stored either in a custom fileformat (see Appendix C.3) or the htseq fileformat, with both being supported for read and write purposes. This allows a user to use htseq generated gene expression counts for the analysis.

The expression values of the individual samples can be visualized in a sortable and filterable table, making it easy to find the interesting regions.

Various information is provided, including the raw sequence count that maps to a particular region as well as the standardized values RPKM [MWM⁺08] and CPM. RPKM stands for Reads Per Kilobase of transcript per Million mapped reads and is a way to normalize the read count across various genes.

The formula to calculate the RPKM of a particular region is:

$$RPKM = \frac{10^9 * C}{N * L} \quad (7.5)$$

Where C is the amount of sequences that map to that region, N is the total amount of sequences for this sample and L the length of the region. As shown in [DRA⁺13], the RPKM value should not be used to compare values between samples, but only to compare the expression of genes inside the same sample. The other normalized value is CPM (Count Per Million) is similar to RPKM but does not consider the length of the region. The formula for CPM can be seen in equation 7.6, where C and N are the same values as for the RPKM formula.

$$CPM = \frac{10^6 * C}{N} \quad (7.6)$$

The CPM value can be used to compare expression levels across samples, but not between regions on in the same sample.

To analyze the data on a sample level, the user can use several filters, such as a name filter, regions of interest filter or a minimal RPKM filter. The data can also be analyzed on a gene, transcript or exon level.

At any moment, additional information about the individual regions can be found through the context menu as well as the possibility to visualize the region in the genome browser.

Once the expression analysis for the individual samples has been done, the differential expression of the genes present in the samples can be determined. We present the tools we implemented for this in the next Section 7.4.2.

Differential expression

Based on the gene expression quantification we implemented a flexible tool that can compare an arbitrary amount of reference and test case samples to determine differentially expressed genes. First the user selects the different samples to be used as a reference as well as test samples. Figure 7.21 shows the setup dialog which also shows configuration options for the comparison.

The first step of the comparison is to normalize the expression values of the different regions across all samples. This normalization step is important, as the amount of sequences counted for a specific region in a sample heavily depends on the total amount of sequences that have been sequenced. To give a sample example, if two sample A and B had the exact same expression of all the genes in them, but sample B would have been sequenced at twice the depth, all genes would show up as differentially expressed. To compensate for this, the counts of all value in sample A need to be double before the comparison.

Various approaches exist to address the problem of normalization and no clear standard has been defined yet. Research papers comparing the different methods are a common occurrence [LGC⁺16, SD13], with no clear best practice yet. Because of this we based our normalization method on the one used in DeSeq [LHA14], which is among the best methods

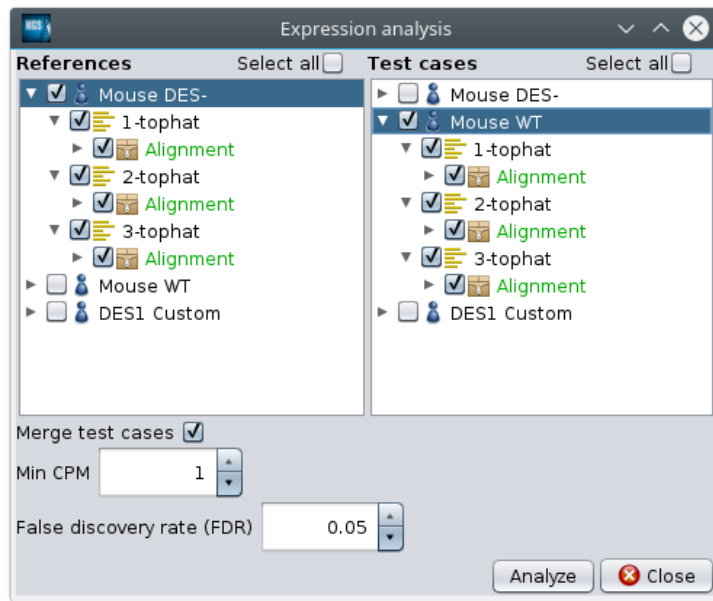


Fig. 7.21.: Setup of reference and test cases for a differential expression analysis

to be used, according to the previously cited publications. We refer to the original paper of DeSeq for the detailed explanation of the normalization method.

Once the normalization of the different samples is done, they can be compared. When comparing the samples on a specific region to determine if it is differentially expressed, we calculate two values. The first is the fold change, which determines how much the expression changed between the two test cases. A fold change of 1 means that there is no difference between the two test cases and a fold change of 2 that the expression of that region has doubled. To calculate the fold change we compare the median values of the reference samples with the median values of the test case samples for this particular region.

The fold change alone, while useful, does not take into account the variance inside the samples and is thus not a reliable indicator if the region is actually differentially expressed. To determine if this is the case, we use a standard *t-test* to determine the probability that the reference and the test cases are differentially expressed. When looking at thousands of regions in the samples, given for example a 5% requirement for the p-value will always give a good amount of false positives. For this reason we also calculate q-value, which is a false discovery rate (FDR) adjusted value, with the goal to reduce the number of false positives. The FDR adjustment is done using the Benjamini-Hochberg procedure [BH95] with a user configurable FDR level.

The resulting list of all analyzed regions is presented in a dialog show in Figure 7.22.

The dialog lets the user filter the results based on his needs, for example to look at a specific gene or only show regions with a given maximum p-value.

Once the list of differentially expressed genes is generated, the user can perform more advanced types of analysis based on the identified differentially expressed genes. Those are presented in the next Section 7.4.2.

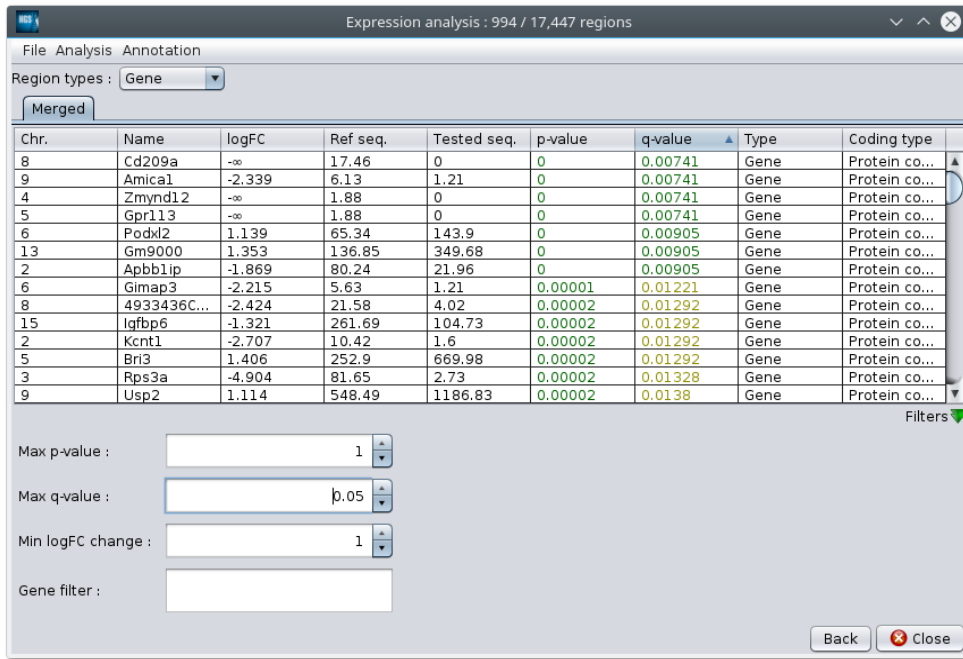


Fig. 7.22.: Result screen of the differential expression analysis

Feature analysis

After successfully identifying differentially expressed genes, a common research question is to determine which biological functions are associated with those genes. Considering all analyzed genes in a sample that are associated with a certain phenotype, it is possible to determine the probability that a certain biological function is affected by the differentially expressed genes.

Various methods exist for this, such as using statistics tests like Fishers exact test or Chi square. We implemented a prototype of a correlation analysis tool, inspired by the method used by GOseq [YWSO10]. Figure 7.23 shows the interface for the correlation analysis.

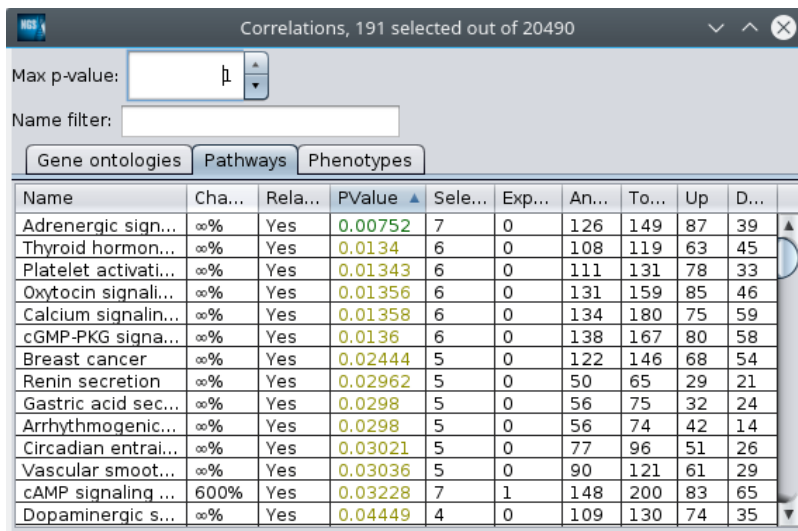


Fig. 7.23.: Correlation analysis interface, showing the relation

The user is presented with the probabilities that a certain gene ontology, pathway or phenotype is differentially expressed.

7.4.3. Summary

In this section we presented a new graphical user interface which allows users to perform RNAseq data analysis based on existing methods. We implemented the tools to perform gene expression analysis as well as differential expression analysis. While existing tools already provide such features, the main goal of this work was to provide an intuitive interface for the user. This interface not only allows to easily perform said analysis, but also integrates transparently with external databases and resource. This helps the user to more easily analyze the results and put them into a biological context.

The resulting tool has been integrated into the graphical NGS data analysis pipeline presented in Chapter 6.

7.5. Epigenetics

This section discusses the NGS data analysis tools that have been implemented to support bisulfite sequencing analysis inside the graphical pipeline discussed in Section 6. Even though the bisulfite sequencing tools have not been released as a stand-alone tool, we choose to dedicate a full section to document them.

7.5.1. Introduction

Bisulfite sequencing is a technique to analysis the methylation of the cytosine (C) nucleotide in the genome. As described in Section 3.3, the methylation of cytosine has regulatory effects on genes, especially related to the methylation (or lack thereof) of the promoter sequence of a gene. While DNA-seq is able to show the genotype of an individual at the DNA level, it cannot describe all biological processes inside a cell. For example, why two genetically identical cells (or individuals) show different gene expression levels. While analyzing the methylation of those genomes might not give an answer to all of those questions, it becomes increasingly clear how important the effect of methylation is on the cell functions.

We used the general DNA-seq workflow developed in the graphical pipeline used for this thesis as the basis for the bisulfite sequencing data analysis. Because of this we focused only on methylation analysis which uses NGS data as its source. As described in Section 3.3.1, the general workflow for methylation analysis and general DNA-seq analysis is very similar. Because of this we decided to add the methylation data analysis as context sensitive additions to the general NGS workflow.

The work integrated into the graphical pipeline regarding methylation analysis is the context sensitive handling of methylation data, both inside the pipeline as well as in the genome browser. In terms of data analysis, the quantification of both CpG and non CpG methylation has been integrated. The quantification is done reference wide as well as in user defined regions, such as the promoter regions of genes. A graphical user interface has been implemented which not only allows for a quick visualization of the methylation of one sample, but also to graphically compare multiple samples. In addition, as described in Section 7.6.4, the genome browser has been enhanced to allow both for easy visualization as well as manual data analysis on the data.

Section 7.5.2 looks at the state of the art of methylation analysis, followed by a more detailed description of what has been implemented in our graphical pipeline (Section 7.5.3).

7.5.2. State of the art

A lot of the bisulfite sequencing data analysis is done using custom R scripts or spread sheet applications. While some stand-alone tools exist like QDMR [ZLL⁺11], wich allows to identify differentially methylated genes and regions, very few DNA-seq pipelines include this type of analysis. The only examples of pipelines integrating it are CLCBio ⁷ and Galaxy [GRH05], with CLCBio providing the most comprehensive data analysis toolset. Stand-alone bisulfite sequence analysis pipelines exists as well, which specialize in only this type of data. Recent examples of such pipelines are MethyQA [SNY13] and SMAP [GZM⁺15], which are automated command line pipelines to analyze the data.

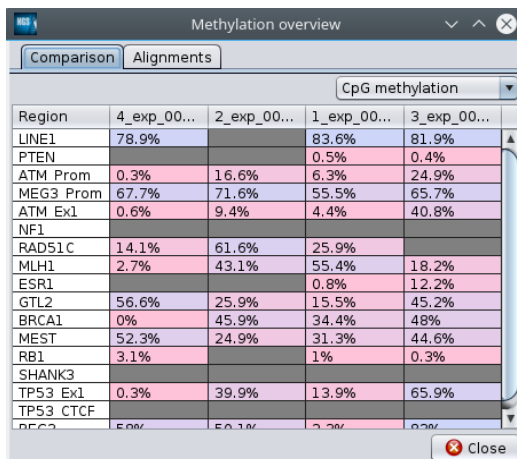
⁷<http://www.clcbio.com/>

Those pipelines generally speaking work through the workflow we laid out in Section 3.3. The different tools used to perform those analysis steps are described in Section 3.3.1.

7.5.3. Methods

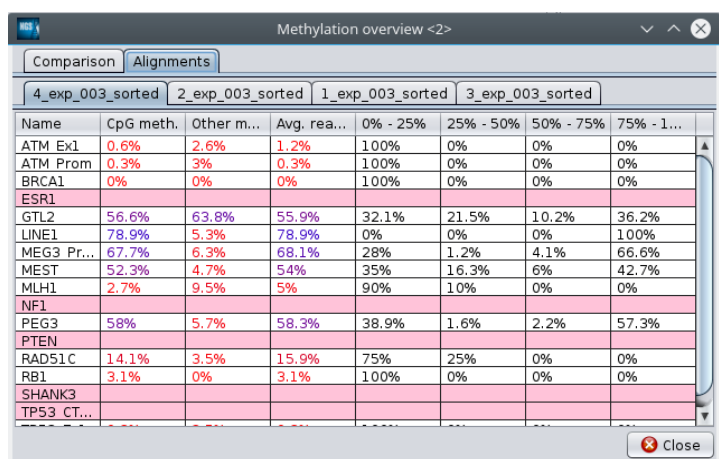
We implemented a quantification tool that determines the methylated reads inside a specific region of an alignment. The results of this analysis are stored in a custom file-format (described in Appendix C.2) to speed up the display for future use. The user can visualize this information for a whole alignment or a specific reference used to create this alignment. It is also possible to visualize more than one sample at the time, to quickly compare the methylation of various samples. Figure 7.24 shows multiple samples which are visualized at the same time.

In addition of the overview information which can be compared between multiple samples, the user can also visualize a more detailed table for the analyzed regions. This is shown in Figure 7.25. The analyzed data as well as its display has been heavily inspired by the needs of the department of human genetics at the University of Würzburg.



Region	4_exp_00...	2_exp_00...	1_exp_00...	3_exp_00...
LINE1	78.9%		83.6%	81.9%
PTEN		0.5%	0.4%	
ATM Prom	0.3%	16.6%	6.3%	24.9%
MEG3 Prom	67.7%	71.6%	55.5%	65.7%
ATM Ex1	0.6%	9.4%	4.4%	40.8%
NF1				
RAD51C	14.1%	61.6%	25.9%	
MLH1	2.7%	43.1%	55.4%	18.2%
ESR1		0.8%	12.2%	
GTL2	56.6%	25.9%	15.5%	45.2%
BRCA1	0%	45.9%	34.4%	48%
MEST	52.3%	24.9%	31.3%	44.6%
RB1	3.1%		1%	0.3%
SHANK3				
TP53 Ex1	0.3%	39.9%	13.9%	65.9%
TP53 CTCF				
BRCA2	58%	58.1%	2.2%	0%

Fig. 7.24.: Multiple samples compared



Name	CpG meth.	Other m...	Avg. rea...	0% - 25%	25% - 50%	50% - 75%	75% - 100%
ATM Ex1	0.6%	2.6%	1.2%	100%	0%	0%	0%
ATM Prom	0.3%	3%	0.3%	100%	0%	0%	0%
BRCA1	0%	0%	0%	100%	0%	0%	0%
ESR1							
GTL2	56.6%	63.8%	55.9%	32.1%	21.5%	10.2%	36.2%
LINE1	78.9%	5.3%	78.9%	0%	0%	0%	100%
MEG3 Pr...	67.7%	6.3%	68.1%	28%	1.2%	4.1%	66.6%
MEST	52.3%	4.7%	54%	35%	16.3%	6%	42.7%
MLH1	2.7%	9.5%	5%	90%	10%	0%	0%
NF1							
PEG3	58%	5.7%	58.3%	38.9%	1.6%	2.2%	57.3%
PTEN							
RAD51C	14.1%	3.5%	15.9%	75%	25%	0%	0%
RB1	3.1%	0%	3.1%	100%	0%	0%	0%
SHANK3							
TP53 CT...							

Fig. 7.25.: Methylation details for one sample

For every sample we collect the amount of detected CpG locations on every region that was analyzed. For our implementation we concentrate on CpG locations, as for humans the CpG methylation is the most important one. We also store the amount of non methylated and methylated CpGs in those regions. Non CpG methylations are also detected and are stored separately, so that during the visualization the CpG and non CpG methylation can be displayed separately. Additionally we detect the average methylation of the reads covering the analyzed section. We also create 4 bins of methylation frequencies between 0 and 100% methylation, to count the amount of sequences that have that amount of methylation.

Every sample which is analyzed can also be easily visualized in the genome browser at the specific location that is indicated in the analysis table. The specific features which have been implemented to both visualize and analyze the data inside the genome browser are described in Section 7.6.4.

7.5.4. Summary

In this section we discussed the integration of basic methylation analysis tools for bisulfite sequencing data into a DNA-seq pipeline. While still in early stages, we built the groundwork

for future methylation analysis improvements in the pipeline. This work on basic epigenetic tools has served as preliminary work for the start of a bigger project in epigenetics in financed by the Swiss Commission for Technology and Innovation (CTI). The project which aims to expand the work done in regards to epigenetics during this thesis started at the end of 2016. While the impact of this work is currently limited, it did already allow various samples to be analyzed and work to be published, as detailed in Section 9.1.4.

7.6. Genome browser

The genome browser is a central tool to analyze NGS data. It allows to visualize and understand the sequenced data as well as to verify the results of an automated analysis. In fact, the first project on which this thesis was started was a genome browser [Wol11], the same which served as the basis for the genome browser presented in this chapter. This chapter is based on published works such as the journal paper *DNaseq Workflow in a Diagnostic Context and an Example of a User Friendly Implementation* [WKDA15a] and the two posters *GensearchNGS-Viewer: A complete NGS data visualization experience* [WKD14b] and *GensearchNGS : Integrating OMICs analysis and visualization* [WKD16c].

7.6.1. Introduction

Genome browsers are a set of tools that allow to visualize information about a genome. They come in various forms, but can be grouped in mainly two groups. Locally installed and web based genome browsers. An overview of various existing genome browsers is given in Section 7.6.2.

For geneticists, genome browsers are an essential tool when analyzing NGS data. It is used for various tasks, but most notably to better understand the sequenced data as well as to verify the results of the data analysis.

Being able to look at the data visually is important to understand it, which is especially important when working with new and unfamiliar datasets, as well as less experienced geneticists. Over the course of this thesis, we observed that once a user doing data analysis gets more familiar with the data, visualizing it becomes less important during the analysis. The visualization is then mostly used to verify the results of the analysis, for example to determine if an identified variant is a genuine variant or a sequencing artifact.

Generally speaking, genome browsers allow to display the alignment of sequenced data, as well as a list of annotations that enrich the aligned data. Sequencing data is stored as a list of sequences with their corresponding positions on the reference sequence. They also contain the alignment information, which indicates how the sequence mapped to the reference at that particular position, especially in regards to indels present in the sequence.

The annotations that enrich the visualization of the alignment data can come in different forms. They can be the locations of the genes on the genome, what protein is encoded by a certain transcript, the location of known variations or many more. Those annotations are critical to understand the data to be analysed, as it puts the aligned data into the biological context.

Genome browsers are not limited to DNaseq data, but depending on the genome browser, different types of NGS data can be displayed. For the genome browser developed during this thesis we mostly focus on DNaseq data, but also include the display of RNAseq and data coming from bisulfite sequencing. While those 3 data-types are similar in many aspects, the display of them is done with different goals in mind, which mostly changes the types of annotations to be displayed when visualizing them. But they also have their specificities, such as the spliced reads of RNAseq, or the nucleotides changes by bisulfite sequencing.

To develop a genome browser which supports those 3 data-types, we based our code on a prototype developed during the master thesis of the author [Wol11]. The genome browser prototype is also discussed in Section 6.2. Before discussing our implementation of a genome browser in Section 7.6.3 we look at the existing genome browsers in the next Section 7.6.2.

7.6.2. State of the art

This section gives an overview at the current state of the art of genome browsers. A more general introduction of the field can be found in section 3.1.1.

Genome browsers can currently be divided into mainly two categories. The locally installed genome browsers which display the content of local data files and the web based genome browsers which work with online data. Both of them have their uses cases and advantages depending on what they are used for. Locally running genome browsers generally provide a much better performance and do not require to go through the slow process of uploaded large datasets to an online server. They also do not pose a data privacy issue when looking at the data, as all the data stays local.

The web based genome browsers on the other hand make it easy to access publicly available datasets and often integrate a variety of data sources which give additional information about the visualized data. The downside is the decreased performance and interactivity, due to the nature of the data to be displayed.

The list of genome browsers discussed in the next two sections is far from complete. Many genome browsers exists, not all of them are being actively developed. This is why we focus on the most used and updated genome browsers for both categories (desktop and web based).

Desktop applications

In this section we take a look at some of the available genome browsers.

IGV: The Integrative Genomics Viewer [TRM13] is arguably the most used locally installed genome browser. Figure 7.26 shows the main window of the genome browser with several tracks.

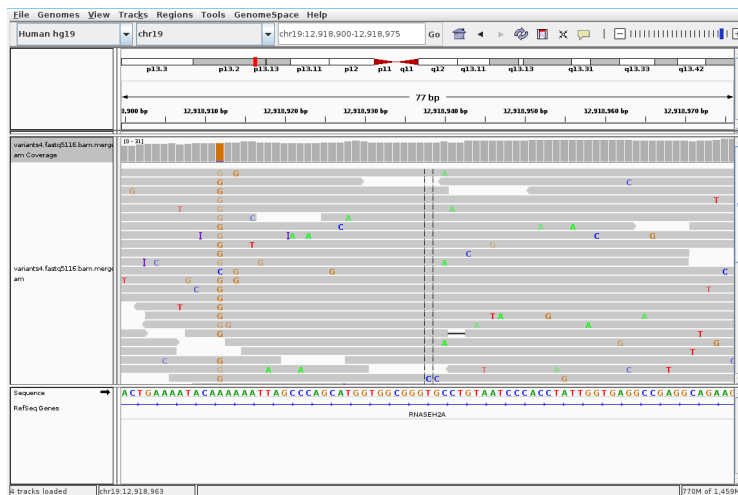


Fig. 7.26.: Example display of the IGV genome browser

The IGV genome browser can work with either locally installed or remote data files, for both the alignment files as well as the annotation files used for the various displayed tracks. One of its main advantages is the big range of supported data formats as well as the ability to stream local and remote content in the most effective way possible. IGV is coded in Java and runs on all major platforms.

Tablet: Another genome browser for NGS data is Tablet [MSB⁺13], which is written in Java just like IGV. Figure 7.27 shows the Tablet genome browser displaying an alignment file.



Fig. 7.27.: Example display of the Tablet genome browser

Tablet is a much more lightweight genome browser than IGV, offering less annotation tracks and integration with external services. For annotation of the sequencing data, Tablet mostly relies on the manual import of GFF files by the user (for example for the gene model to be displayed). On top of standard alignment files, such as BAM files, Tablet has also been developed to visualize de-novo sequence assemblies.

Web applications

Several web based genome browsers exist. They are very popular and widely used, among other reasons because of their integration of annotations which are also available online. Indeed, two of the most used genome browsers, the UCSC genome browser as well as the Ensembl genome browser, are produced by communities which offer a wide range of genome annotations.

We now take a look at some of the available web based genome browsers.

UCSC genome browser: This web based genome browser allows the user to display a variety of genome annotations, as well as alignment files, made available by the University of California, Santa Cruz. Figure 7.28 shows a screenshot of the UCSC genome browser [KSF⁺02].

Through a very complete set of configuration options, this genome browser allows to integrate a variety of annotation files and display them at the same time using several tracks. Those annotation files are either provided by the UCSC project, or can be provided by the user. fact

Ensembl genome browser: Very similar to the UCSC browser, the Ensembl browser [CAB⁺15] is also based on a genome annotation service which provides a variety of annotations. Indeed, we use the Ensembl service as our main data source for genome annotations. Figure 7.29 shows a typical screenshot of the Ensembl genome browser.

Similarly to the UCSC genome browser, the Ensembl genome browser allows the display of various annotations in several track, with a lot of configuration options on how to display

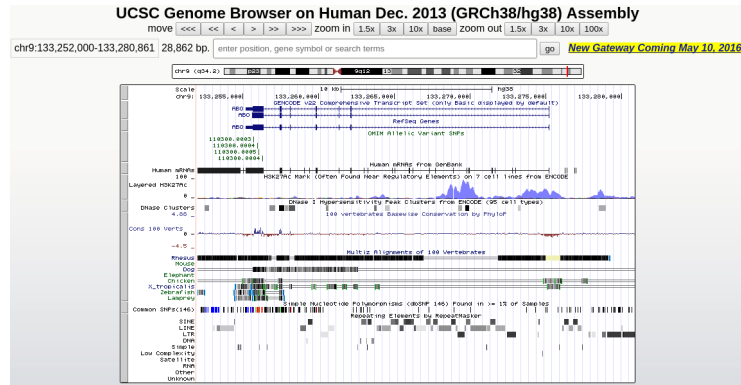


Fig. 7.28.: Example display of the UCSC genome browser

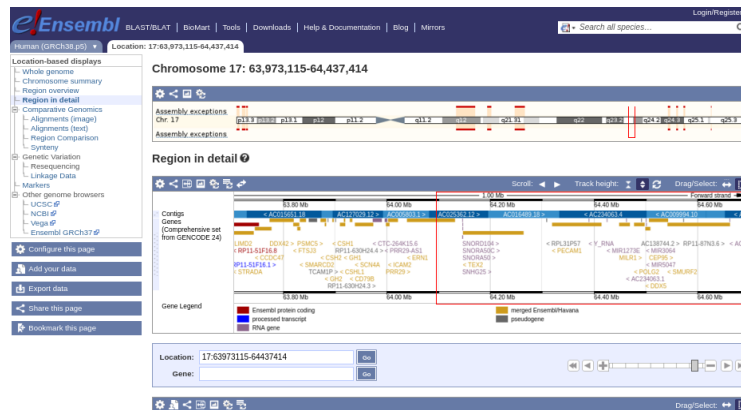


Fig. 7.29.: Example display of the Ensembl genome browser

them. The main difference with the UCSC browser is the direct integration with the Ensembl annotation database, which removes the need for many annotation features to be loaded from files.

JBrowse This genome browser takes a slightly different approach to the two previously mentioned web based genome browser. This genome browser is based on HTML5 and JavaScript to create a flexible and extensible experience for its users. But instead of being directly integrated into a larger platform, JBrowse [BYD⁺16] is an independent genome browser, which is intended to be integrated in various websites.

Figure 7.30 shows the JBrowse genome browser with multiple active annotation tracks.

Compared to other previously presented genome browsers, JBrowse focuses on being integrated into existing websites and being extensible. Unlike the UCSC and Ensembl genome browser, the JBrowse website does not provide an usable genome browser installation out of the box. The JBrowse software has to be integrated into an existing website and extended by plugins to match the websites needs.

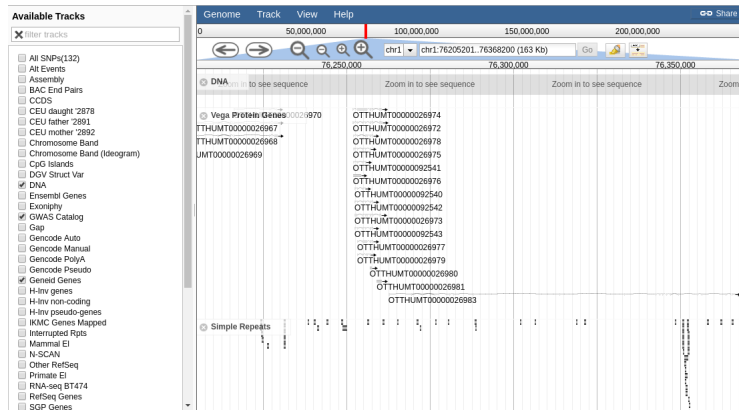


Fig. 7.30.: Example display of the JBrowse genome browser

7.6.3. Methods

Based on the specific needs in diagnostics for a well integrated genome browser, we decided to create our own. This allows us to have a maximum flexibility on how it operates and to better adapt it to the needs of the geneticists.

The genome browser we developed uses the previously developed prototype of a genome browser, which was written in Java. It can be either executed as a stand-alone environment using local or remote data, or as an integrated part of the visual pipeline developed during this thesis (see Chapter 6).

The genome browser can read data from various data sources. For the alignment data, BAM files are supported through the Htsjdk⁸ library, which is part of the samtools project [LHW⁺09]. The BAM files organize the aligned sequences by their alignment position, which makes it possible to quickly extract the information of a specific region to be visualized.

The reference sequence is read from a FASTA file using a custom data reader. The FASTA file can either contain only one reference (for example for one chromosome), or multiple references (for example all chromosomes in the same file). To quickly access the correct reference in the files, FASTA index files are used. If not already present, the genome browser is able to create a new index file automatically.

The annotations displayed in the genome browser can come from a multitude of sources. They are grouped into two groups, the integrated and the external annotations.

Both of those annotation categories are very similar, as they add additional information to the alignment to give the user a biological (or other) context when looking at the data. But with internal annotations, we know what they represent and can thus be displayed in a more integrated manner. External annotations are displayed “just” as graphs, which can contain arbitrary data for which we do not know the meaning.

The integrated annotations are mostly used for the gene information provided by the Ensembl [YAA⁺16] project. The way annotation data is recovered and managed from Ensembl is shared between the genome browser and the graphical pipeline, and described in more detail in Section 6.6. The annotations from Ensembl provide the user with the information about the location of the genes, as well as the different transcripts of the gene. Based on this information, an interactive display of the protein encoded by the gene as well as the consequences of the variants on that protein can be displayed. This information is crucial when doing diagnostics and the impact of a variant on a particular gene has to be identified.

⁸<http://samtools.github.io/htsjdk/>

Other annotations come from external files, where multiple types of data-formats are supported. Those include BED, WIG and GFF, which are used to display different types of annotations. All three of those file formats are text files which contain information based on the position on the genome. BED (Browser Extensible Data) ⁹ files are commonly used to mark certain regions of the genome, be it genes or repetitive regions. WIG ¹⁰ files are used to store graphs associated with the genome. GFF (General Feature Format) are mostly used to store gene information (their positions and exons). The support for those data-formats is based on custom code written during this thesis.

As far as annotations go, we mention variation data separately. The variation data can come in the form of VCF [DAA⁺11] files as well our custom format (initially inspired by the Varscan 2 file format) described in Appendix C.1. The support for those file formats is based on custom code which was created according to the needs of the application, which is to say, fast access and easy integration in the existing codebase.

Other annotations, like the GC content of the reference, are calculated dynamically while displaying the data.

The different data-sources have been programmed in a way that abstracts the actual data-format from the underlying data. This makes it possible to replace a file format with another for almost any source, making the application future proof if the standard file formats change.

To optimize the genome browser, VisualVM ¹¹, the official Java profiler was used.

The following Section 7.6.4 shows the different features which were integrated into the genome browser to support NGS data analysis.

7.6.4. Results

This section presents the features implemented in the genome browser for the different data types to be visualized. Based on the needs of the users, which have been determined through various use cases (see Chapter 9 for examples), many features for the visualization of alignment data have been integrated.

First we look at Figure 7.31 which gives an overview of the genome browser with various features.

We can see several tracks displayed in that figure, showing different features of the data. Starting from the top, the first track gives an overview of the currently displayed gene. We can see the different exons (blue) as well as the currently displayed portion of the gene (gray). The user can freely click around this overview track to quickly navigate to different parts of the genome, or in this case, the selected gene.

Just under the overview track we can see the gene track which shows the location of all genes. In this case, as we are zoomed in to a specific gene, only one gene is displayed, but if the user zooms out further, all genes of the currently visible region are displayed.

A very useful track is displayed just under the genes track, which is the protein track. The protein track displays the protein encoded by the currently selected transcript. The top display in the protein track shows the protein encoded by the reference, and the protein displayed just under it is the one encoded by the consensus sequence of the currently displayed alignment. The user can also at any moment select a particular base in an aligned

⁹<https://genome.ucsc.edu/FAQ/FAQformat.html#format1>

¹⁰<https://genome.ucsc.edu/goldenPath/help/wiggle.html>

¹¹<https://visualvm.java.net/>

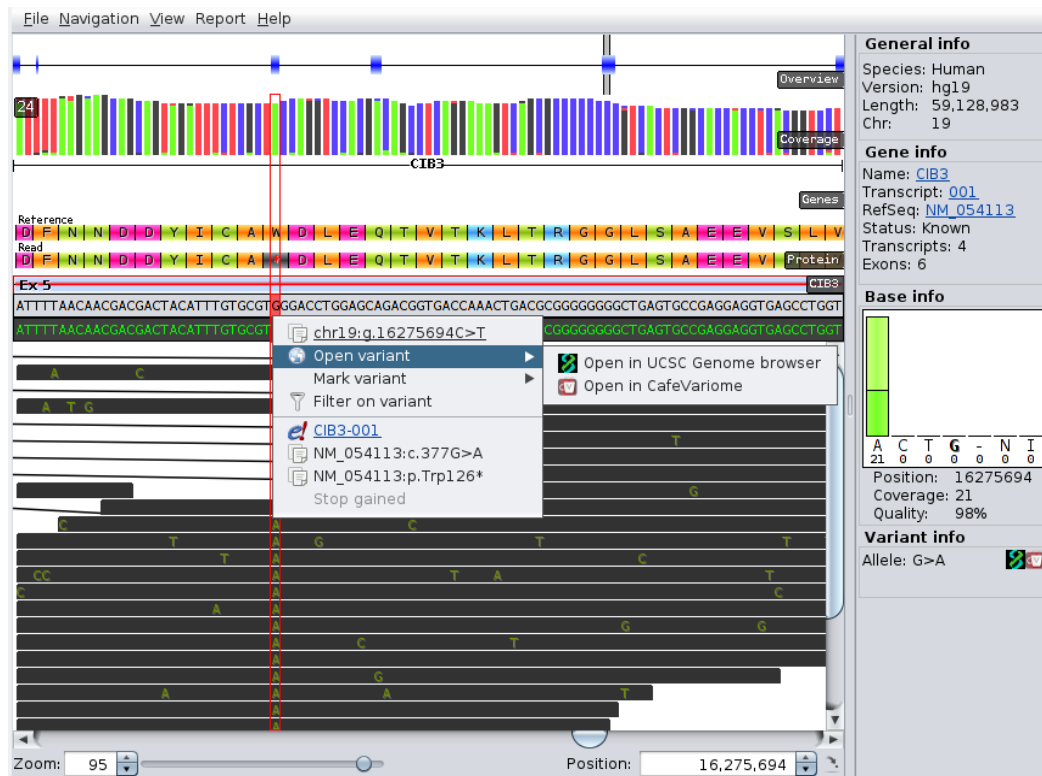


Fig. 7.31.: Main view of our genome browser, showing multiple annotation tracks

sequence, to see its consequence on the encoded protein. This allows the user to quickly evaluate the changes made by the variants found in the data on the currently displayed gene/transcript.

The next track shows the currently selected transcript in more detail. This track also shows other transcripts for the gene, if available, and lets the user switch between the currently selected transcripts.

Next up are the reference and the consensus sequence track. Both of those tracks show a DNA sequence, with one being the reference sequence and the other the consensus sequence of the current alignment.

Right clicking on a variant (as shown in the screenshot), will open a contextual popup menu displaying various actions to the user. This includes the possibility to open the variant on external websites, like the UCSC genome browser.

Just under the reference and consensus sequence the actual sequence alignment is shown. Different visualization methods for the display of the reads have been implemented, with the default one being shown in the screenshot. By default, all forward and backward reads all displayed in the same area, with only the nucleotides displayed that differ from the reference. The displayed nucleotides are colored by their quality information (as given by the sequencer) and the sequence backgrounds are colored by their alignment quality (as determined by the aligner).

On the right side, various information about the currently displayed data as well as the selected gene is displayed. We can also see the histogram displayed on the right, making it easy for the user to quickly see the nucleotides present at a given position.

The following sections will look in more detail at the various features in the genome browser.

Annotation tracks

As shown in Figure 7.31, our genome browser contains various annotation tracks. Those tracks can either come from traditional annotation files, like BED, WIG and GFF files, but also from the internal data-structures which are used to display the data. Those internal data-structures are for example the alignment data, information derived from the alignment data (like coverage) as well as gene and transcript data coming from Ensembl.

Here is a list of the various tracks available in our genome browser:

Overview: The Overview graph, also shown in Figure 7.31, gives the user the ability to quickly see what portion of the selected gene he is visualizing. Not only does it show the displayed parts and where the various exons are, but it also serves as a way to quickly navigate the alignment data. By clicking on any region on the gene on the overview view, the user can jump to that particular position.

Coverage: The coverage track shows the coverage of the alignment data at any given position. The coverage is shown with color colored bars, based on the standard colors of the different nucleotides, which are, green for A, black for C, red for T and blue for G. Those colors are also user configurable in the genome browser configuration. When zoomed in, this allows the user not only to quickly evaluate the coverage of the alignment, but also see if there are many positions with heterozygous variants. The coverage information is directly calculated from the visualized alignment data when zoomed in close enough. When the user zooms out more, for example to the level of the full genome, a precomputed coverage information is shown. The reason why the coverage needs to be precomputed when visualizing large parts of the alignment is for performance reasons. To do this, we developed a custom binary data-format that allows for a quick retrieval of the coverage data for visualization purposes.

Quality: Using the quality track, the user can get a graph which shows for every position in the alignment its quality, which is defined by the average quality of all aligned nucleotides at this position. This allows the user to quickly evaluate the sequencing quality of various positions in the alignment, in particular around variants.

Reference GC content: The information on the GC content in the reference is an important information for the user, as the GC content can influence the quality of the sequencing. The effect the GC content can have on sequencing is well known [BS12] and can cause problems when analyzing certain parts of the genome with a particularly high GC content. The GC content of the reference refers to the amount of G and C nucleotides in a specific region of the reference. The GC content is the percentage of those bases found in relation to the other two bases, A and T. The exact formula to calculate this can be found in 7.7, where r is the GC content in percentage.

$$r = \frac{G + C}{A + C + G + T} \quad (7.7)$$

By default, the genome browser calculates the GC content using a sliding window, with a window size of 21. This value is configurable by the user.

Genes: The genes track allows the user to see all known genes (as they were retrieved from Ensembl) and their location on the genome. This makes it possible to see if a certain region of the alignment is on a gene or not.

Transcript: The transcript track shows by default the 5 biggest, known, protein coding transcripts. This definition of what a known transcript is comes from Ensembl, novel and experimental transcripts are two other states a transcript can have. In the transcript track, the user can select the currently active transcript, which will be used for all information that needs an active transcript, like the protein track discussed next. The transcripts show the individual exons, as well as the UTR regions, in distinct colors. This makes it possible to quickly see what part of the transcript is touched by a certain feature, like a variant.

Protein: The protein track shows the encoded protein by both the reference sequence, as well as the alignment data. For the alignment data, by default the consensus sequence is used. If the user wants to see the effect of a particular variant on the encoded protein, he can select the variant and the protein will be updated, making it possible to visually compare the reference and encoded protein.

View modes

When looking at sequence alignment data, not all users need the exact same information. Depending on the analysis being done, different information is important.

By default, all aligned sequences are shown side by side, positioned at their aligned position. Paired end reads are drawn on the same level, connected through a line, highlighting their relationship. The reads are colored based on their sequence alignment quality, with lighter colors indicating a low quality alignment and darker colors higher quality alignments. This allows the user to quickly see if a certain region contains mostly high or low quality alignments, which might indicate duplicate regions like pseudo-genes. The individual nucleotides which make up a read are hidden by default, and only shown if they differ from the reference sequence. The displayed nucleotides are colored by their quality, which is determined by the sequencer, indicating how sure the sequencer is that the nucleotide is indeed correct. The color green is used for high quality nucleotides and the color red for low quality nucleotides. Both of those colors can be changed by the user.

The behavior of the alignment display can be changed by the user. The biggest change the user can make is to separate the forward and backward reads. This will duplicate the sequence display area, and only show reads aligned on the forward strand in one, and only reads on the reverse strand on the other. This allows the user to see if there is a mismatch in aligned sequences in both direction, which could indicate sequencing problems of the particular region.

Alternatively to quickly see in what direction the aligned sequences are in a particular region, the user can change the coloring of their background. Instead of using the quality information for the background, the direction of the sequence is used. Reads on the forward strand are then displayed in blue and reads on the reverse strand in orange.

When displaying paired end data, it is also important to know how the distance between both read pairs, as well as the information if a read is paired at all. To quickly see this information, the user can choose to color the background of the aligned sequences based on their distance to their pair. The higher the distance, the lighter the color, with darker colors indicating closer distances to the sequence pair. Reads that are not paired, are displayed with an orange background.

To help the user look at the data he is interested in, various filters have also been implemented. Two types of filters have been implemented specifically for DNaseq data. Align-

ment quality and read pairing filters. The first filter, alignment quality, allows to filter reads based on both minimum and maximum alignment quality. As sequence aligner assign an alignment quality based on the probability that a sequence is aligned at the correct position. This is especially useful to filter out sequences with an alignment quality of 0, which is commonly used to indicate that a particular sequence can be aligned at multiple positions equally well.

The other filter that was implemented is the read pair filter, which allows the filtering of sequences which are either paired or not paired. Often unpaired reads are discarded as they may indicate a quality issue of the particular read, leading to artifacts in the data.

The architecture to implement those filters is very flexible and can easily be extended based on user needs. Having interactive filters integrated in a genome browser is from our knowledge a feature only found in our genome browser.

Different data-types require different visualization and filters. This is why a special view mode as well as filters have been added when integrating bisulfite sequencing data. More details about that can be found in Section 7.6.4.

Data reports

While the main purpose of the genome browser is the exploration of sequence alignment data, an important part is to create reports for the findings made using the genome browser. Directly integrated into the genome browser we added the ability to create various reports based on the data currently being visualized. Those reports include the ability to create a report based on the variants identified as well as the coverage of the visualized region.

Figure 7.32 shows an overview of the average coverage per exon of the currently selected gene that the user can display. In Figure 7.33 we see the minimum percentage of bases covered at certain depths.

The coverage information displayed is based on the currently selected gene and active transcript. This allows the user to easily create statistics of the transcript of interest and visualize them. As in both figures, the user can also change a variety of settings to adapt the statistics to his needs. This includes the possibility to include or ignore the untranslated regions (UTR) of the gene as well as set a margin before and after every exon to be taken into account. This information is crucial for quality control of the sequenced data, for example to identify low coverage exons in which variant calls might be unreliable.

The other main information displayed by the genome browser for which the user needs to create reports are the variants. The user can annotate, filter and visualize variants coming from either a VCF file or our custom variant file format. Once a variant has been analyzed by the user, he can mark the variant using various labels which follow the ACMG guidelines [RAB⁺15] to later save them in a report. The handling of the variants is shared between the genome browser and the graphical pipeline presented in Chapter 6. The details of how variants are handled (including annotation, filtering etc.) is discussed in Section 6.7.4.

Once the user marked one or more variants as being interesting, he can create a report that can be exported to either an HTML file or a PDF document. The PDF document is created by converting the created HTML document to a PDF file using `cutycapt`¹². The usage of `cutycapt` is optional and only proposed to the user if the software is installed.

¹²<http://cutycapt.sourceforge.net/>

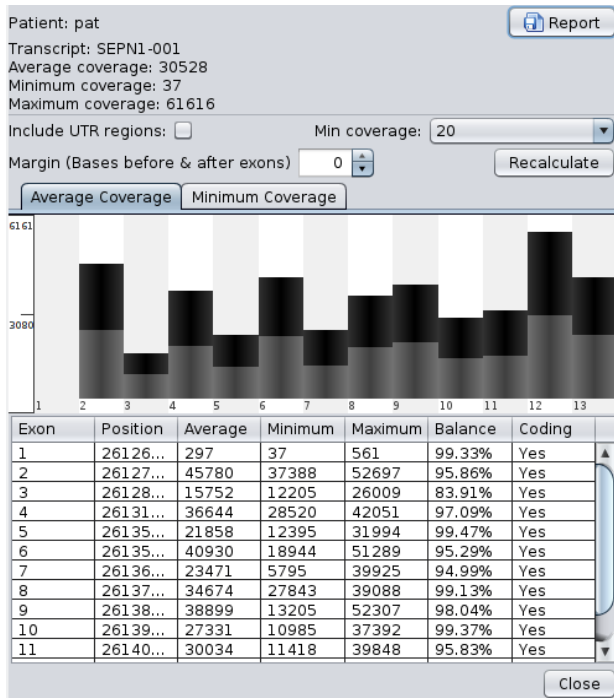


Fig. 7.32.: Average coverage of all exons of a transcript

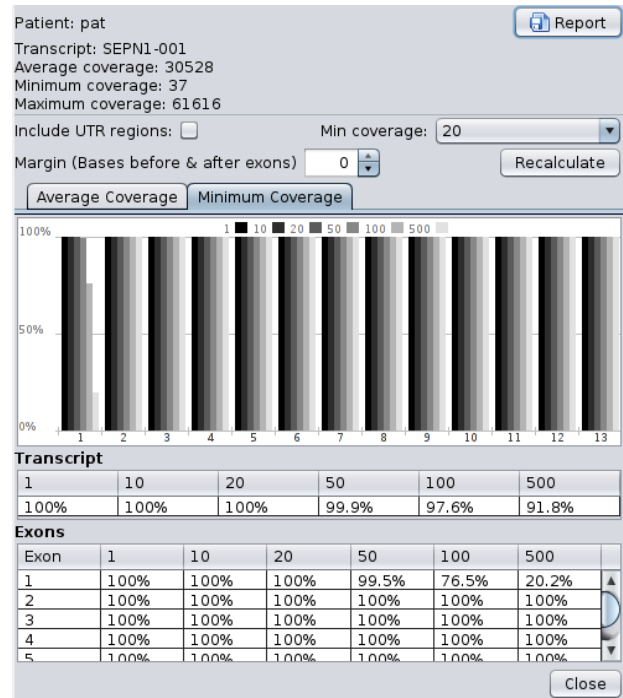


Fig. 7.33.: Minimum coverage of all exons of a transcript

RNA-seq

Visualizing RNA-seq data is very similar to DNA-seq data. Both have very similar underlying data, with sequences that are aligned against a reference. The main difference is that RNA-seq aligners often perform so called spliced alignment. Spliced alignment can introduce large gaps in an aligned sequence, which behave similarly to deletions, but do not count as such (and are usually much longer). Those splices are introduced to map RNA sequences, which follow the exons of a transcripts, onto a DNA reference sequence, which also contains the introns not present in the RNA. Figure 7.34 shows an example of spliced reads, while at the same time showing the possibility of displaying NGS data from a mouse genome.

Introducing support for RNA-seq reads mostly consisted in allowing for those spliced reads. Special handling of splices inside of the sequences had to be introduced, to make it possible to visualize the alignment without increasing memory requirements as well as having good performance.

As for the rest of the features needed for RNA-seq, they were mostly already supported by the needs of DNA-seq visualization. This includes the possibility to easily see the exact position of a particular (or multiple) transcript as well as their exons. This allows the user to visualize the relationship of the aligned RNA sequences the corresponding transcripts.

Methylation

To support methylation analysis and in particular bisulfite sequencing data, several features have been integrated to support the use-cases of researchers using this feature. The features introduced were mostly based on the requirements of the human genetics department at the University of Würzburg, which lead to several works discussed in Section 9.1.4.

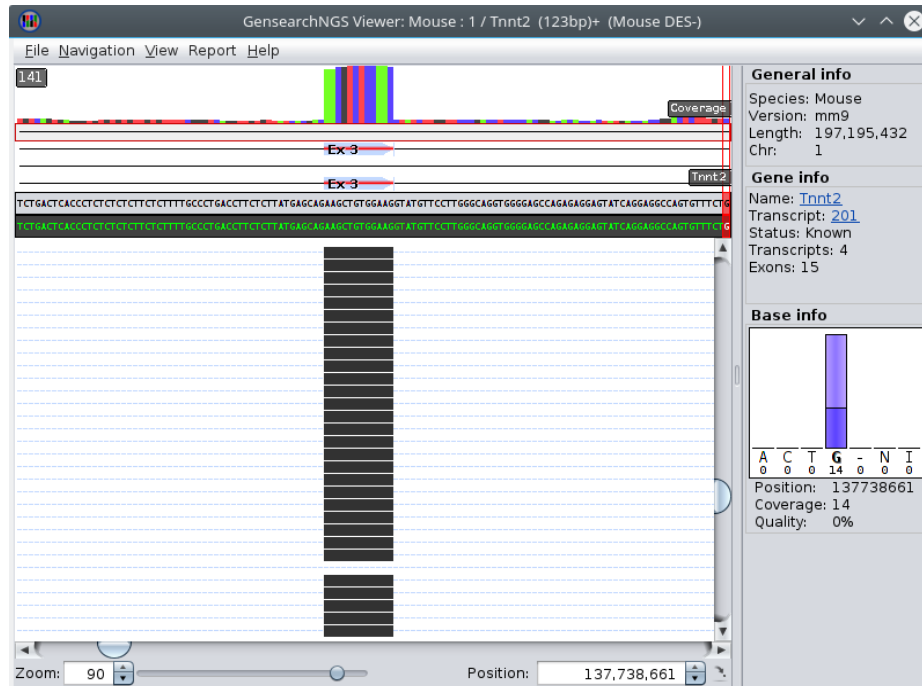


Fig. 7.34.: Spliced RNAseq reads displayed in the genome browser

Methylation analysis, which is part of epigenetics, studies the differentially methylated C (cytosine) nucleotides. Methylated C nucleotides are chemically modified cytosine molecules which influence various biological processes in the DNA, notably gene expression. To detect those methylation events in the DNA, through a chemical process called bisulfite sequencing, the non-methylated C nucleotides are transformed to T (thymine) nucleotides. When sequencing this treated DNA, the remaining C nucleotides in the sequenced reads are the methylated C nucleotides.

Sequencing data like this can be done directly with no modification of the genome browser, but with the downside that the visualized data is very noisy. Indeed, all non-methylated C nucleotides will be shown as T nucleotides, which look like variants when visualized.

We extended the features in the genome browser to display bisulfite sequencing data in a more comprehensible way. If bisulfite sequencing data is treated as normal NGS data, the genome browser will display all non-methylated Cs (which have been transformed into Ts), as variations. Because of this, we changed the color of all non CpG related C to T conversions, making it easier to visually identify the relevant C to T conversions. Additionally, all occurrences of the C nucleotide in aligned reads, which indicate either a methylated C or a problem in the bisulfite conversion process, are highlighted in yellow. The CpG positions in the reference are also highlighted, to make it easy for the user to have a quick overview of where the potentially methylated positions are.

To have a quick overview of the methylation status of a certain position, a new graph was introduced which shows the methylation percentage of the reads covering the currently selected position. The same information is also available on a read level, when selecting a specific read. Using those features, the user can identify the methylation level of a specific region through the genome browser.

We also added additional filters, making it possible for the user to filter reads with a maximal or minimal amount of methylation. Using the flexible filter backend, we also added

filters to specifically filter reads based on their non CpG methylation amount, as well as requiring a certain minimal amount of CpG places on the read. Using those filters, the users are able to manually explore the methylation level of specific regions in the alignment.

To solve a different use case, which is the separate methylation analysis of paternal and maternal DNA in the same sample, we added a novel filter to the genome browser. In cases where a variant is specific to either the paternal or maternal side, the user can create a filter to only show reads which either contain or do not contain a specific variant.

Miscellaneous

The list of features is much larger than this document could hold, this section has a short overview of features that were not mentioned in the previous sections but are still interesting to note.

One of the most important features of a genome browser is the ability to navigate the genome. In our browser, this is done through various means. To visualize bigger or smaller regions of the genome, the ability to zoom has been integrated. This can be done either through a slider or a numerical input, as seen in Figure 7.31.

When wanting to change the current region to be displayed, various options exist. The user can navigate using the scrollbar at the bottom, which also gives him an overview of where in the genome he is located. This is directly linked to an input field where the exact position to jump to can be specified. To navigate only with the keyboard, the arrow keys can be used, with the CTRL key modifier allowing to move faster through the data.

But the user also has the possibility to navigate using the various annotations, notably the overview track and the transcript track. Using the overview track clicking on any region of the overview jumps to that particular region. With the transcript track, one can jump to a specific exon number through a right click menu.

Another way is to jump directly to a specific variant using the variant browser, which is discussed in more detail in Section 6.7.4.

Last but not least, a searchable list integrated in the genome browser make it possible to select any of the genes present on the reference and visualize them. When selecting a gene, the area to be displayed is restricted to the extents of the gene, making it easier to concentrate on the gene currently being visualized. The different ways to navigate correspond to the different use-cases the users of the software have, making it possible for them to navigate optimally for their use-case.

The ability to select a gene, as well as an active transcript, brings us to another important aspect of the genome browser, which is context sensitive information. To not overwhelm the user, whenever possible only the information relevant to the current context is displayed. An example would be the transcript track, which is only visible when a gene is selected, and similarly the protein track which is only useful when a transcript is selected. Selecting a gene also switches all data displayed to the direction of the gene. For genes which are on the reverse strand, this allows the user to view them just like any other gene. This behavior can be changed by the user.

The contextual information also relates to variants and the information that is associated with them. The right click menu shown in Figure 7.31 shows the ability to open that particular variant on various external websites. Again, only the relevant websites are shown for a particular variant. In the case of the variant in the Figure, because there is no ID associated with the variant, only generic websites are displayed. Other plugins which are integrated allow the user to open the variant in Ensembl [CAB⁺15], Clinvar [LLB⁺16],

Cosmic [FBG⁺15], Exac [Lek15] in addition to the UCSC genome Browser[KSF⁺02] and CafeVariome[LBA⁺15]. A special plugin is the Alamut ¹³ plugin, which opens a variant in the variant analysis software Alamut, developed by interactive software. Alamut offers the user various metrics and annotations to evaluate the impact of a variant.

This integration of external websites also applies to genes and transcripts, allowing the user to open the relevant information on Ensembl and other websites.

The information shown and linked has been based on user feedback and continual adaptation to their workflows.

7.6.5. Summary

We presented a genome browser which visualizes DNaseq, RNAseq and bisulfite sequencing data. The focus has been put on ease of use and the integration of external resources, to give proper context to the visualized features. This allows the user to easily navigate and understand the genomic features he is visualizing. It was crucial, that the genome browser properly integrates with the workflow of both diagnostics and research environments. This was not possible using existing genome browser, as they are designed to work as stand-alone applications. Having a complete control over the genome browser, allowed us to implement the specific visualizations and features needed to efficiently analyze NGS data in a diagnostics or research environment. It also helped to tightly integrate the genome browser into the graphical pipeline developed in Chapter 6.

7.7. Discussion

This chapter presented several works done to improve NGS data analysis.

We explored distributed sequence alignment and the possibilities it offers to better use existing infrastructure. This with the goal to lower the complexity of maintaining a complex computing infrastructure to handle NGS data. We could show that using a flexible distribution architecture, it is possible to adapt to a variety of infrastructures. This allows to combine multiple local machines, as well as grids and private or public clouds to accelerate the data analysis, depending on the specific needs and restrictions (e.g. privacy issues). Future developments should expand this approach to allow existing aligners, such as BWA etc. to use this approach and not be limited to our own implementation of an aligner.

The new method of meta-alignment presented in this chapter approached the problem of the complexity of choosing the right aligner for the right situation and dataset. By combining the output of multiple aligners, we are able to compensate for the weak points in the various aligners, while at the same time benefiting from their strong points. We showed that the meta-alignment approach is especially interesting for complicated datasets, e.g. datasets with a high amount of errors. We also showed that the approach can be used to either improve the alignment rate or the precision of the resulting alignment file. After the promising initial results of this approach, future developments need to address the current shortcomings of the method, such as the lack of support for paired end sequencing data.

After the sequence alignment, for which we presented two works, the resulting alignment files are often scanned for variants. Upon noticing that this analysis step could take a similar amount of time as the alignment itself, we proposed an optimized variant calling method,

¹³<http://www.interactive-biosoftware.com/alamut-visual/>

reproducing the same results as an already existing variant caller. By speeding up the variant calling processes by up to 18, we were able to reduce the total analysis time significantly, as well as to give the user the possibility to quickly rerun the analysis using different analysis settings. The proposed architecture for the optimized variant calling was flexible enough to also be used for other use-cases, such as coverage analysis. Due to the modular and flexible nature of the variant calling we developed, we intend to improve the variant calling by integrating other variant calling methods that work in parallel, improving variant calling in a similar approach to how our meta-alignment approach improved sequence alignment.

For RNAseq analysis we developed a new intuitive user interface that integrates into the graphical pipeline discussed in Chapter 6. We based our approach on existing methods while integrating external databases to make it easier to analyze the complex datasets. We had a similar approach for bisulfite sequencing data, where we developed a user interface for the most important types of analysis. Both of those approaches were validated and used through their usage on real projects, discussed in Chapter 7.6. For future development, the features implemented need to be expanded to more precisely respond to the needs of both research and diagnostics.

Visualizing complex data such as NGS data is very important, as it helps to more easily understand the data as well as to visually verify the results of automated analyses. This is why we implemented a custom genome browser that is adapted specifically to the needs of diagnostics and research. Various features were implemented based on user feedback and tightly integrated into our graphical pipeline. While already a large amount of external databases are integrated to supplement the visualization with relevant biological information, this is also the point where in the future most progress can be made to help users to analyze NGS data. Information like conservation scores across multiple species or the display of multiple alignments simultaneously could help to more easily determine the biological impact of a given genomic feature.

8. POP-Java

In this chapter we present the development of the POP-Java (Parallel Object Programming) language, a Java extension which makes it easier for programmers to write parallel distributed applications. We present the state of the art of the field, the state of POP ecosystem when the project started as well as the improvements implemented to get it to its current state. We also present new ways in which the POP model is used, such as the cloud, Hadoop clusters and the distribution of calculations over a network of friends. We evaluated the POP-Java approach in Section 8.8. Parts of this chapter have been published in different publications, such as *POP-Java : Parallélisme et distribution orienté objet* [WKD14c] (POP-Java : Object oriented parallelism and distribution) and *Distributed programming using POP-Java* [WK13b]. The first sections of this chapter describe the existing work that was already done on the POP model and the POP-Java language. The author's contributions are discussed from section 8.5 onward.

8.1. Introduction

Developing distributed and parallel applications has a long history (see Chapter 5), but up to this day it remains a difficult concept to grasp and implement. With the general theme of our research being to explore how the complexity of OMICS analysis can be lowered, this includes research on how to lower the required computing infrastructure. We showed earlier (Chapter 7.1), that parallel and distributed computing can help to accelerate NGS data analysis and achieve this goal.

It is very common that a laboratory has multiplay available computers. Most of the time, those computers are idle, while other computers are pushed to their limit. Combining their resources by distributing the computing workload more evenly among them seems like an obvious solution, but it is rarely used due to its complexity.

To make the development of such distributed applications easier for programmers, something which still greatly hinders the development of distributed applications, a new distributed programming approach was needed. We wanted to both make it easier to develop and maintain such applications, as well as to setup and maintain the required infrastructure to run it.

A natural starting point was the POP (Parallel Object Programming) paradigm, which was initially developed as a PhD thesis by Tuan Anh Nguyen [NGU04]. One of the thesis supervisors was Pierre Kuonen, which also supervises this thesis. While the POP model does indeed make it easier to develop parallel and distributed applications, its initial implementation POP-C++ [NK07] is not suited for this thesis, which is done in Java. Because of this, an initial POP-Java prototype, developed by Valentin Clement [CS10], based on POP-C++ [NK07], was used as a starting-point.

In the rest of this chapter we present at the state of the art of parallel computing, distributed computing and language extensions. While not exclusive to the Java world, we will mainly concentrate on methods related to Java.

In this chapter we then discuss the improvements brought to the initial prototype, leading to a publicly released open-source tool. We will also discuss new concepts brought to distributed computing, like friend-computing as discussed in Section 8.7.3.

8.2. State of the art

This section discusses the state of the art of distributed computing in Java and other programming languages in general. It also addresses the subject of existing Java programming language extensions, what they do and how they integrate into the existing Java language. This section is puts the author's work described in later sections into context.

8.2.1. Parallel computing in Java

Parallel and distributed computing have a long history in Java. Traditionally, programmers used *Threads* to perform concurrent computing in Java. The synchronization was done using manual locks and semaphores which could lead to easily missed errors. This problem lead to the development of the JSR-166 (Java Specification Request 166) ¹ which included a list of improvements of the Java standard library in terms of parallel computing. Without being complete, the introduced changes included atomic variables, improved parallel collections and thread handling. Those improvements where included in Java 5, reducing the complexity of writing parallel applications for the programmer.

The latest improvement to Java are the introduction of lambdas and parallel streams which allow the user to more easily write map and reduce like algorithms which are executed in parallel. Those changes were introduced in Java 8.

As discussed in Section 5.3.2, over the course of recent years, GPGPU had a great influence on parallel programming. Being able to access thousands of computing cores on a local machine, even if individually less powerful than a CPU, allows for great speedups in many applications. Of course the Java world did not ignore this movement, and also integrates the power of GPUs through various approaches.

To integrate GPU computations into the Java programming languages, there are basically 2 approaches. One being the integration directly into the Java Virtual machine, the other being library based or wrappers around CUDA or OpenCL. We will briefly present the different available approaches in the rest of this section.

Project Sumatra ² is a project which intends to bring the power of GPUs directly inside the Java virtual machine. Through the use of Java 8 Lambdas, parts of the Java application will be transparently offloaded to a GPU, without the programmer having to write specific code. This project is still in an early development phase, but has a high potential. It is also worth mentioning that IBM is working in parallel on a different implementation of the same concept ³, which further shows the potential of this approach.

Aparapi ⁴ is a library based approach to the integration of GPU offloading in the Java language. Aparapi generates CUDA code at runtime for parts of the application where the

¹<https://jcp.org/en/jsr/detail?id=166>

²<http://openjdk.java.net/projects/sumatra/>

³<http://blogs.nvidia.com/blog/2013/09/22/gpu-coming-to-java/>

⁴<https://github.com/aparapi/aparapi>

programmer used the Aparapi library. An OpenCL backend is also supported, but even more experimental than the CUDA implementation. A very similar library to Aparapi exists, called **Rootbeer**⁵, but it is at the time of the writing of this document no longer actively developed.

JCuda⁶ uses a different approach and is essentially a wrapper around native CUDA code. The programmer writes the code to be executed on the GPU using standard CUDA code in a C file. A very similar library, *JavaCL*⁷, exists, but using OpenCL instead of CUDA. Just as JCuda, JavaCL lets the programmer code in “native” OpenCL instead of relying on a compiler to translate Java bytecode into a code executable on the GPU. For both projects, this allows the programmer to reuse existing code written in CUDA or OpenCL to integrate them into a Java application.

Often simply using all the available processing power on a single machine is not enough. In that case, multiple machines are connected, using a distributed computing approach as presented in the next section.

8.2.2. Distributed computing

Many libraries support development of distributed applications in Java. One of the more commonly used ones is even integrated into the standard library of Java, called RMI (Remote Method Invocation)⁸. RMI allows the programmer to publish locally running objects on the network, and other instances of Java applications can connect to those objects to use them. RMI does not handle concurrent access to the exposed objects, any access is handled the same way as multiple local threads access any Java object, with the programmer having to assure thread safety of the methods. To use RMI objects on a remote computer, the objects need to be created first by that computer through an already running Java application. The programmer cannot directly instantiate a new object on a remote computer.

A similar approach to RMI has been taken by DIRMI⁹, which is a mostly source code compatible replacement for RMI. It promises a better performance as RMI as well as bidirectional connections, which allow objects to be accessed through a NAT, which is not possible with RMI. It also introduces the concept of asynchronous methods for remote objects, which is not present in RMI.

Concepts like the actor model [HBS73] for distributed computing are also present in Java. A multitude of libraries implement that model using different approaches. One example, out of many is Kilim [Sri10], which implements a high performance implementation of actors for distributed computing. The Akka¹⁰ project should also be mentioned, which is probably the most famous Java and Scala actor model framework. Both of those actor model frameworks draw heavy inspiration from the Erlang [CT09] programming language.

One of the more commonly used distributed programming models in the Java world, at least for large scale problems, is Hadoop [Whi12], which implements the MapReduce [DG08] model. Developed by the Apache foundation, a rich ecosystem developed around Hadoop, which lately became very popular for large scale problem solving and big data applications.

⁵<https://github.com/pcpratts/rootbeer1>

⁶<http://www.jcuda.org/>

⁷<https://github.com/nativelibs4java/JavaCL>

⁸<http://www.oracle.com/technetwork/java/javase/tech/index-jsp-138781.html>

⁹<https://github.com/cojen/Dirmi/wiki>

¹⁰<http://akka.io/>

The distributed computing frameworks mentioned in this section largely integrate into the Java programming language as libraries. The next section discusses how the Java programming language can integrate features not initially intended, through extensions of the programming language.

Last but not least, ProActive [CHS04], a Java middleware which distributes objects over multiple computers using a special job scheduler. ProActive uses the concept of active objects to allow for distributed and parallel programming within Java. In contrast to POP-C++ and the original POP-Java, it is not a language extension, but a middleware which provides the programmer with a set of tools and libraries to achieve its goals.

8.2.3. Language extensions

This section presents the different ways to extend the Java programming language to integrate new features. This does not include simple libraries, but more advanced techniques to achieve things not possible through normal means or changes to the programming language itself.

One of the first examples to extend the Java programming language is as well the most extreme one. Java source code is compiled to Java byte code. The Java byte code is machine independent code and is executed through the Java virtual machine, in short JVM. This assures the ability to run the same code on multiple platforms, as long as there is a JVM available on the platform that can run the Java byte code.

The Java byte code is independent to the Java programming language, it is in many ways comparable to assembler. This means, it is possible to create different programming languages than Java, which compile to Java byte code and benefit from the JVM running on various platforms. There are several well known examples of this type of approach, with the most famous one being Scala [Oa04], a functional programming language developed for the JVM. Groovy¹¹ and Clojure¹², are two other examples of this approach. Groovy is a JVM language that intends to combine the Java syntax with the Ruby syntax, and can be used both as a compiled programming language like Java and a script language, which is compiled only at runtime. Clojure on the other hand is a LISP dialect developed for the JVM, which through its syntax and functional programming approach best shows how flexible the JVM is to accommodate different programming language concepts.

A very different approach to the previously mentioned, but which has a similar end result, is to create a pre-compiler, which takes source code and transforms it into Java source code. This can be used to integrate small additions to the Java programming language, like the initial POP-Java prototype did [CS10], or even create completely new languages. But in that latter case, the approach to directly compile to Java byte code is more common, as one is not limited by the features of the Java language, but only the Java byte code.

The third approach, which is used by projects like Lombok¹³ is to use internal APIs of the Java compiler to integrate new features into the Java programming language. While this approach is technically risky, as undocumented internal APIs can break with every minor release of Java, it offers many possibilities. The Lombok project for example adds new key keywords to the Java programming language, like *val*, which is a generic type converted to the actual type only at compile time.

¹¹<http://www.groovy-lang.org/>

¹²<https://clojure.org/>

¹³<https://projectlombok.org/>

The last example of Java language extensions comes through the use of Javaagents. In Java, a Javaagent is loaded before all other classes and is able to modify the byte code of classes to be loaded through the use of libraries like Javassist¹⁴. This can be used to create wrappers around methods, for example to write a profiler that measures the time it takes to call a method, or more advanced changes to the language. One example is the detection of integer overflows at runtime, which is done by the project COJAC¹⁵ using this technique.

8.3. POP Model

The core principle of the POP model [NK02] is that objects are ideal candidates to distribute data and computation over the network. The author did not extend the POP model, but used it to implement the Java language extension POP-Java.

In object oriented programming, objects encapsulate data as well as functions. The POP model makes it possible to distribute the execution and storage of those objects over multiple computers and lets them interact between each other. This creates an intuitive approach for programmers already familiar with object oriented programming to develop distribution applications. The POP model transparently distributes the objects, letting the programmer interact with them almost as if they were “normal”, not distributed objects. The programmer can for example instantiate an object which calculates the number π , which is then automatically created on a remote computer. When calling the method of the object which does the calculation, it is transparently executed on the remote computer.

One key problem to solve when distributing objects in a distributed and parallel context, is the order of execution. While in a local, sequential environment, the order of execution of multiple method calls on an object is well defined, this is no longer the case in a distributed environment. In a purely local but parallel environment, programmers need to handle those problems with locks, semaphores and other concurrent programming tools. With one of the main goals of the POP model being the reduction of complexity of programmers to program distributed systems, the programming model itself integrates the tools required to handle parallel access to an object.

POP objects define a set of public or private methods as well as private attributes. Due to the distributed nature of the objects, public attributes are not possible in the model.

A public method in a POP object contains in its definition how it is accessed in a parallel environment. Two attributes are defined for every public method. The first is to define if a method is *synchronous* or *asynchronous*. In most programming languages methods are synchronous, meaning that the caller waits for the method call to end before continuing its own execution. With asynchronous methods, the caller immediately continues to execute his code. The ability to use asynchronous methods allows the programmer to transform a sequential application into a parallel application.

The second attribute every public method possesses, handles the execution order in the case of multiple parallel method invocations on the same object. In the POP programming model, methods can follow one of three possible execution orders. The most basic one is called *concurrent*, which indicates that the method can be called in parallel with no restrictions (it could also be called thread safe). The second one is called *sequential*, which assures that all calls to methods of an object with the attribute sequential will be executed

¹⁴<http://jboss-javassist.github.io/javassist/>

¹⁵<https://github.com/Cojac/Cojac>

in order of arrival, but not in parallel. It has to be noted that a sequential method can be executed in parallel with a concurrent methods.

The last option is called *mutex*, which indicates that a method must run exclusively, with not other method of any type running at the same time. This means that once a method call on a mutex method arrives, it has to wait for all current calls finish, and all calls which come in during the mutex call have to wait for it to finish before execution.

Through the use of those attributes, which create six possible method configurations, the programmer can program parallel and distributed applications through a much simpler approach than the commonly used threads and other tools. It is also from a programming stand point the same if an object is run in parallel on a second CPU on the local computer, or on a remote computer.

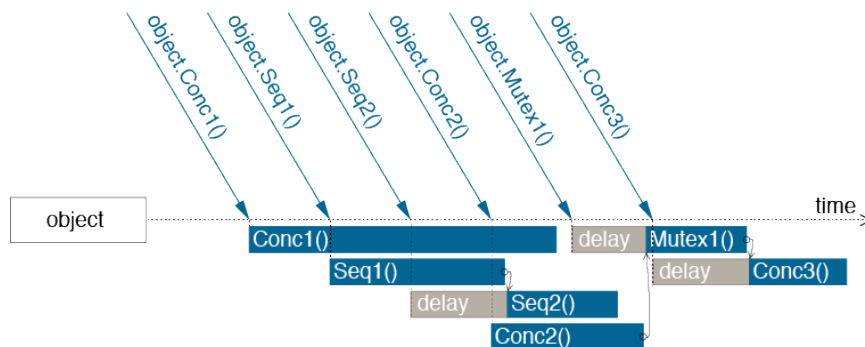


Fig. 8.1.: Method call ordering, taken from the official POP-Java manual

Another interesting property of POP objects, even if implicitly assumed, is that they can be passed around by reference. Just like in a traditional programming language, objects can be parameters for method, and thus be passed around. Whenever a POP object is sent as a parameter to a method of a POP object, the reference to the original object is passed around, allowing the receiver to transparently connect to the original object.

One important thing to note is that the POP model does not have shared memory between multiple POP objects. All communication between POP objects is done through their public methods, which is why as previously stated, public attributes are not allowed on POP objects.

8.4. POP Java prototype

In 2010, an initial prototype of POP-Java was developed by Valentin Clement [CS10]. It was based on POP-C++ version 1.3.1 beta J1 [NK07], to the point that both implementations are compatible and are able to share objects between each other. The prototype demonstrated the feasibility of bringing the POP model to the Java programming language, but had its limitations which are discussed in Section 8.4.1.

In the initial prototype the POP-Java programming language, several keywords were added to the Java language, similar in style to the POP-C++ language. Those keywords were integrated into normal Java source code, with the files using it having a new file extension `.pjava` instead of the traditional `.java`.

Listing 8.1 shows an example of how POP-Java code looked like in the prototype.

Listing 8.1: Example POP-Java code from the prototype

```
public parclass Example{

    public Example(String url) @od.url(url); } {

    public sync conc int getValue(){
        return 42;
    }
}
```

We can observe that essentially three changes had been made to the Java language. At first, the addition of the keyword `parclass`, which marks a particular class as being a POP-Java class. This is required, because POP-Java objects can exist in the same application as normal Java objects, and thus need to be marked as such. The second change is the addition of the object descriptors, used in the constructor. The object descriptors allowed the programmer to configure certain aspects of the object to be created, such as the IP of the machine on which the object should be created (as seen in the example). The last addition was were the keywords `sync`, `async`, `conc`, `seq` and `mutex`. They were placed in the method signature of all public methods of a POP-Java object, and thus defining the access policy to the method.

As seen in Section 8.2.3, there were three ways to integrate those keywords into the Java language. For the initial prototype, the approach with a pre-processor was used, which transformed the POP-Java code into regular Java code which could then be compiled by the Java compiler. For this, JavaCC¹⁶ was used, which allows to define a syntax grammar, and based on an input file, create a corresponding output file. The parser was based on Java 6 with the addition of the POP-Java keywords.

The next Section 8.4.1 discusses the limitations of this initial prototype, followed by Section 8.5 which addresses those shortcomings.

8.4.1. Limitations

The initial prototype had a list of problems which did not make it suitable for wider usage. The first problem was the way the POP model was integrated into the Java programming language. As discussed earlier, the prototype added several keywords to the Java language. Those keywords made the source code of POP-Java applications incompatible with normal Java tools, especially the Java compiler and IDEs. Not being able to use the modern tools which assist programmers when developing POP-Java applications, severely lowered its usefulness.

The second problem was the requirement to go through a custom parser, which translated the POP-Java annotated source code into standard Java. POP-Java was limited to the functionality of Java 6, as the parser was specifically written for that version of the Java language. Every new feature added in the Java language required updating the custom

¹⁶<https://javacc.java.net/>

parser, a non trivial task, especially when thinking about features like lambda functions added in Java 8.

The third problem of the prototype was that it was in fact a prototype, with many big and small issues bugs and design problems. Various core features did not work correctly, such as the keywords like `sync` and `conc`, which made the usage of the language impossible in real life scenarios. On top of the various issues in the core functionalities, there were also various performance problems which limited its usefulness in performance critical applications.

While those problems appeared rather serious, with some fundamental design choices being questioned, the core of the POP-Java language was nevertheless a solid base to build future versions upon. The next section presents at what the improvements were that have been brought to the POP-Java programming language and how the described downsides were addressed.

8.5. Implementation

Based on the initial prototype described in Section 8.4, a new version of POP-Java has been developed, which not only improved the existing concept to make it usable in real life projects, but also introduced new concepts. This section presents the author contributions to the POP-Java language.

Section 8.5.1 discusses how the POP-Java language was transformed from a keyword based Java language extension, to an annotation based language that is compatible with existing Java tools. In Section 8.5.2 the additional changes to the POP-Java code are discussed.

8.5.1. Annotations

The major downside of the initial POP-Java prototype was the usage of a custom pre-processor which not only made maintenance complicated to adapt to new Java versions, but also made it impossible to use standard Java tools, like IDEs.

To move away from hard coded keywords which are not part of the Java programming language, to a pure Java based approach, it was decided to use Java annotations. The design and implementation of the move to an annotation based system has been done by the author.

Listing 8.5.1 shows how POP-Java code looks like after this changes have been implemented.

Listing 8.2: Example POP-Java code implementing an Integer class, source [WKD14c]

```
@POPClass
public class Integer {
    private int value;

    @POPObjectDescription(url = "localhost")
    public Integer() {}
    public Integer(@POPConfig(Type.URL) String url) {}
}
```

```

@POPSyncConc
public int get(){ return value; }

@POPAsyncSeq
public void set(Integer i){ value = i.get(); }

@POPAsyncMutex
public void add(Integer i){ value += i.get(); }
}

```

We can see the new code compared to Listing 8.1, providing the programmer with a more familiar Java approach to program in POP-Java.

In Java, annotations can be added to classes, constructors, methods and even parameters. They allow programmers to add additional information to the source code, information which can be used by the compiler or other tools. All POP-Java keywords from the initial prototype were converted in a first step to annotations. Table 8.1 gives an overview of the different annotations and how they related to the keywords of the prototype.

Tab. 8.1.: Relation between the old prototype keywords and the new annotations

Keyword	Annotation	Description
parclass	@POPClass	Marks a class as being distributed
@{ }	@POPObjectDescription	Defines various static parameters when calling constructor
@{ }	@POPConfig	Annotates constructor parameters for dynamic object configuration
IN/OUT	@POPParameter	Defines serialization settings for method parameters
sync conc	@POPSyncConc	Defines method as synchronous concurrent
sync seq	@POPSyncSeq	Defines method as synchronous sequential
sync mutex	@POPSyncMutex	Defines method as synchronous mutex
async conc	@POPAsyncConc	Defines method as asynchronous concurrent
async seq	@POPAsyncSeq	Defines method as asynchronous sequential
async mutex	@POPAsyncMutex	Defines method as asynchronous mutex

Most keywords can be recognized from the POP model description. The IN/OUT keyword is a special keyword that is used to optimize POP applications. By default all parameters to a method are synchronized when calling the method, and again sent back to the caller after the method finished. This is done to keep the behavior of local method calls even in a distributed environment. When for example a Java method has an integer array as a parameter, and this array is modified inside the method call, it will also be modified for the caller. This behavior is not always desired in a distributed environment, especially if the method parameters are large (such as a matrix). For this reason, the programmer can decide if a parameter is IN (it is only sent the called method, and not sent back), OUT (it is not set to the called method, but sent back) or by default IN/OUT.

The annotations replace, as discussed earlier, the pre-processor which converted POP-Java code into real Java code. To achieve this, a `javaagent` was implemented which intercepts all class loading events in the JVM, and changes the bytecode of the loaded class if it is a POP-Java class. The bytecode manipulation is done using `Javassist`¹⁷, which has two main functions. The first is to let all POP-Java classes be extend the `POPObject` class. In contrary

¹⁷<http://jboss-javassist.github.io/javassist/>

to the Java programming language, the JVM does actually allow for multiple inheritance on a byte code level, allowing for this type of byte code manipulation. The second function is to replace all object instantiations of a POP-Java object by the corresponding call to the POP-Java utility function which correctly instantiates a remote object.

Thanks to those changes, POP-Java code is regular Java code that can be edited with any Java IDE, as well as used with traditional Java tools for development. It also made POP-Java independent of the Java version used by the programmer.

8.5.2. Additional changes

The initial POP-Java prototype had several problems, discussed in Section 8.4.1. Fixing the underlying problems, such as the bugs that made proper usage impossible was the first priority. In addition to extensive bugfixing, the build system was overhauled and moved to *Ant*, away from *make*, as it is more appropriate for Java based projects.

In addition of the integrated testsuite, which tests a variety of features that POP-Java offers using real applications, a unit test suite was integrated. Using JUnit, more lightweight tests were added, allowing for automated testing during development.

The whole documentation system was also overhauled, moving from a latex based documentation to one based on *Read the docs*¹⁸. This move allowed to generate not only a printable PDF for the manual, but also an interactive web based manual that could be integrated into the POP-Java website. The online manual is automatically kept up to date by tracking the commits on the public POP-Java git repository, thus reducing the manual work required to keep the documentation up to date. The manual can be found on the official POP-Java website¹⁹.

Another architectural change made to POP-Java was to make it possible to integrate POP-Java functionalities into an existing Java application. Using the old system, which was similar to how POP-C++ works, the main class of the application needed to be a POP-Java class. This was required to assure that all functionalities of POP-Java were initialized correctly. This was changed, so that an existing Java application can instantiate a POP-Java object in any part of the code, without having to convert the complete application first. This was achieved by introducing a system where the POP-Java environment is initialized as soon as the first POP-Java object is created. Through this, the POP-Java functionalities are only be started if required and do not require a complete rewrite of the original application.

Various other changes have been made, which range from bug fixing, optimizations to the compatibility with the latest POP-C++ version 3.0.

8.6. Usage examples

In this section we discuss two examples of how the POP-Java language can be used to help distributing applications over multiple machines. We discuss how POP-Java makes the development of the example applications easier and look at their performance where appropriate. All the examples presented are contributions made by the author. The first section 8.6.1 looks at a more theoretical use-case, the multiplication of matrixes in a distributed environment to evaluate the performances of POP-Java. The second section 8.6.2 discusses

¹⁸<https://readthedocs.org/>

¹⁹<http://gridgroup.hefr.ch/popjava/>

the usage of POP-Java in an example that performs parallel and distributed calculation of the mandelbrot set.

8.6.1. Distributed matrix multiplications

Multiplying big non sparse matrices is a common task in many scientific fields. While the goal of this thesis is not to create an efficient matrix multiplication implementation (many projects exists for that matter), it does lend itself for a good example to show how POP-Java behaves in a distributed environment.

In this section we discuss how POP-Java scales over multiple machines and how it helped to implement this type of calculation. In 8.6.1 we discuss the implemented algorithms and improvements made to POP-Java to be efficient at this task. We also show code examples on how POP-Java was used to implement the algorithm. The results of the benchmarks are discussed in Section 8.6.1.

Methods

To test the performance of POP-Java in a different context than sequence alignment, we choose to implement a matrix multiplication algorithm of two square non sparse matrices. The goal of this test is to test the object creation speed, the data transfer speed as well as the speedup over multiple computers. It also served as a test-case to improve POP-Java for this type of calculations.

We implemented the standard matrix multiplication algorithm with a complexity of $O(N^3)$, where every cell of the resulting matrix is calculated by 8.1.

$$R_{ij} = \sum_{k=1}^n A_{ik} B_{kj} \quad (8.1)$$

This algorithm was implemented in a `Matrix` class, handling the storage of the matrix. To allow for optimal data access, two versions of the class have been implemented, `Matrix2DCL` and `Matrix2DLC`.

The actual calculations are implemented in a class called `MatrixWorker`, which can be distributed. Listing 8.3 contains a simplified version of that class, showing off the different features of POP-Java used to implement that class.

Listing 8.3: Code example used to implement the matrix multiplication with POP-Java

```
@POPClass
public class MatrixWorker {
    ...
    public MatrixWorker(@POPConfig(Type.URL) String url, int
        cores, int minStrassen) { ... }

    @POPAsyncSeq
    public void solve(@POPParameter(Direction.IN) Matrix2DLC a,
        @POPParameter(Direction.IN) Matrix2DCL b) {
        MatrixWorker me = PopJava.getThis(this);
```

```

        for (int i = 0; i < cores; i++) {
            me.solveCore(i);
        }

        for (int i = 0; i < cores; i++) {
            calcSemaphore.acquire();
        }
    }

    @POPAsyncConc
    public void solveCore(int core) {
        for (int j = core; j < a.getLines(); j = j + cores) {
            for (int k = 0; k < b.getCols(); k++) {
                double r = 0;

                for (int l = 0; l < a.getCols(); l++) {
                    r += a.get(j, l) * b.get(l, k);
                }

                result.set(j, k, r);
            }
        }

        calcSemaphore.release();
    }

    @POPSyncMutex
    public Matrix2DCL getResult() { ... }
}

```

While most features have been described earlier, we can observe one particularity of that class. In order to achieve both distributed and local parallelism, the class has two methods related to the matrix multiplication, `solve` and `solveCore`. The first one, `solve`, is the main function, which is called by remote objects. It would be possible to perform the complete matrix multiplication inside this method, only one core of the machine would be used. In order to use multiple cores, the `solve` method calls `solveCore`, which calculates one specific part of the matrix multiplication. As the `solveCore` method is asynchronous and concurrent, it can be called multiple times in parallel, thus resulting in parallel execution on multiple cores of the machine. To achieve this, the `solve` method first retrieves a POP-Java reference on itself (the `me` variable), which allows the object to call its own methods, while following the POP method call conventions. By default method calls inside the same POP-Java object follow the standard Java method call conventions. After having executed the multiple asynchronous method calls on `solveCore`, the `solve` method waits for them to end. This is done by using a standard semaphore, which is unlocked once the individual `solveCore` calls are finished.

This allows the programmer to program an algorithm that behaves similarly to multi-threaded algorithm, without having to manually create and manage the different threads.

This approach also shows that we can use POP-Java easily for both local and distributed parallelization.

Once the matrix multiplication was implemented, we tested its performance (in particular in regards to scaling). The test environment used is the same as for the sequence alignment 8.8.1, which consists of 10 quad-core machines with 8 GB of RAM.

Results

We tested the matrix multiplication with varying (square) matrix sizes, using 1-10 computers to perform the multiplications. The tested matrix sizes are 840, 1680, 2520, 3360, 4200, 5040, 5880 and 6720. To test the speedup, we only measured the time for the actual calculation, not the time required to transfer the matrices over the network.

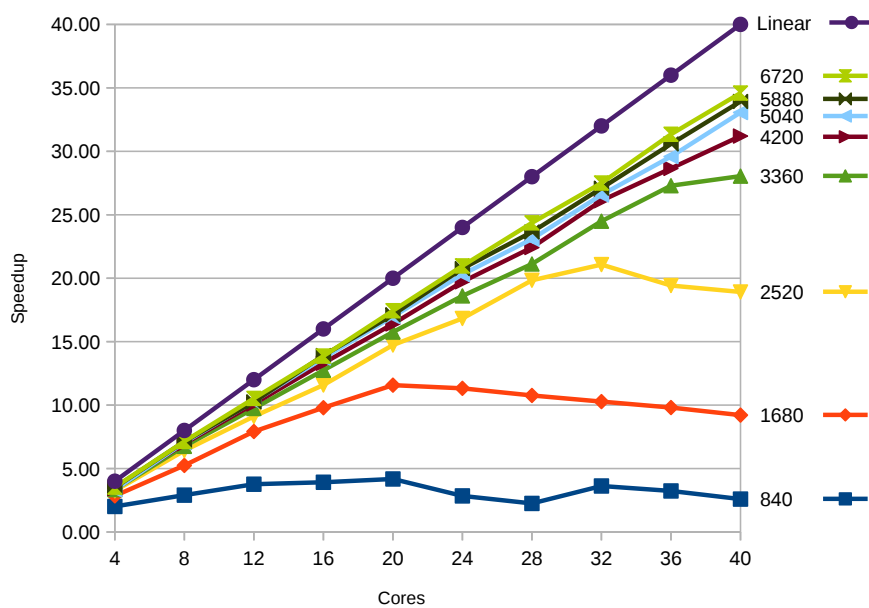


Fig. 8.2.: Matrix multiplication speedup

We can observe that the speedup of the multiplication is very dependent on the matrix size. The speedup results are good, which shows that we can handle parallel computations efficiently when using POP-Java.

We deliberately did not show the overall speedup that includes the data transfer times. This is because the ratio between data transfer times and calculation times is quite bad in the case of matrix multiplications. This is why in the next section, we looked at an algorithm where much less data is transferred compared to the time required for the calculations.

8.6.2. Distributed mandelbrot

As shown in the previous example of matrix multiplication (Section 8.6.1), the performance of distributed algorithms is very dependent on the ratio of transferred data to computation. To show an example of a distributed algorithm which requires less data to be sent around compared to the computation that needs to be done, we implemented a mandelbrot set visualization algorithm in POP-Java. An example of this visualization (which was created using our test-code) can be seen in Figure 8.4.

The mandelbrot set is a fractal which is constructed using the formula 8.2:

$$z_{n+1} = z_n^2 + c \quad (8.2)$$

This computationally intensive fractal can be used to produce beautiful images, but has also inspired mathematicians to study it extensively, namely in the context of the chaos theory. Section 8.6.2 gives an overview of how the mandelbrot visualizer was implemented and Section 8.6.2 discusses the benchmark results.

Methods

We implemented a simple mandelbrot visualizer which can output a visualization of the mandelbrot set both to a GUI or a file. The workload was split by dividing the image to be produced into horizontal strips, with every computer calculating the content of one of the strips. The different computers in the grid received each one POP-Java object, which also handled the multi-core computing of the part of the image they received. In the same way the original image is split among the computers, the image every computer calculates is split into multiple sub-images for every core to handle. Without the programmer having to use threads, we distributed the workload over multiple computer and their cores.

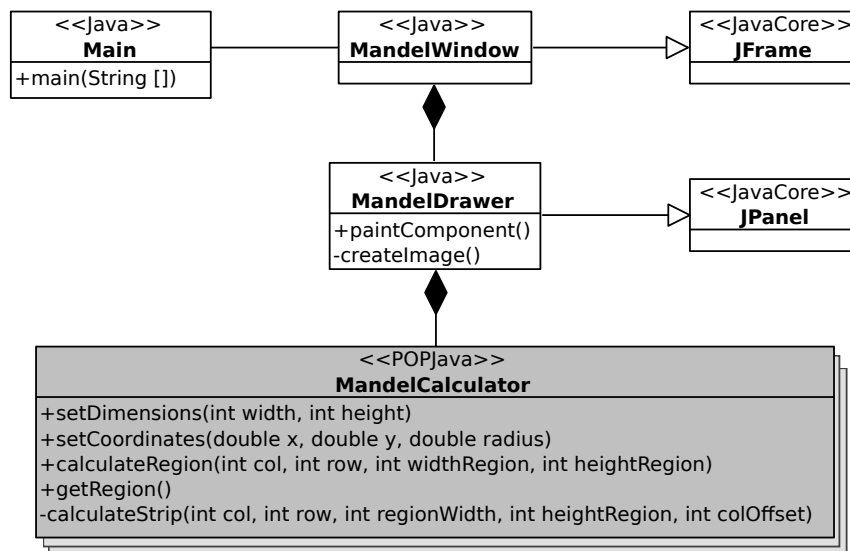


Fig. 8.3.: UML diagram of the Mandelbrot calculator

Figure 8.3 shows the UML diagram of the Mandelbrot test application. The only POP-Java object in the application, is marked with a gray color. This diagram also shows, how POP-Java and normal Java objects can easily interact with each other.

To test the efficiency of calculation we distributed the calculation over 1-10 computers using the same setup as the distributed sequence alignment 8.8.1 and the matrix multiplication 8.6.1.

Results

Figure 8.5 shows the speedup of the calculations over 4-40 cores (1-10 computers) using different image sizes.

We can observe how the speedup is dependent on the image size, reaching much better speedup values than the previous example of matrix multiplication (Section 8.6.1). The

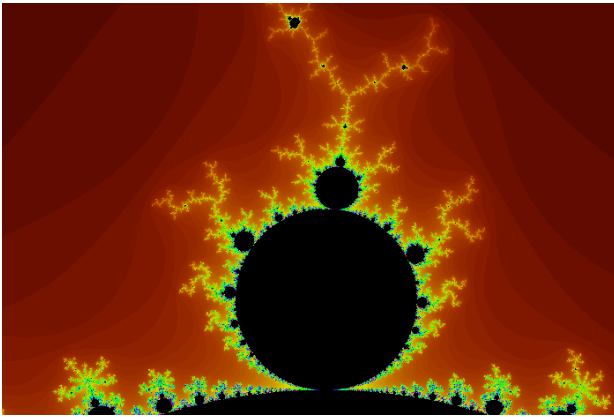


Fig. 8.4.: Visualization of the mandelbrot set with our application

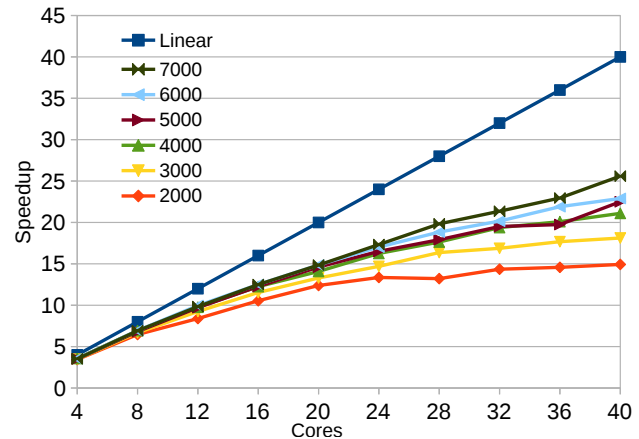


Fig. 8.5.: Speedup of mandelbrot calculations over 10 computers

provided graph shows the speedup of the complete calculation, which includes data transmission times. Compared to the matrix multiplication example, almost no data had to be transmitted to the remote objects to start the calculations. The only data which had to be transmitted was the resulting image which had to be retrieved from the individual remote objects. Again, like the distributed matrix multiplication, the additional code needed to distributed (and/or multithread) the calculation, is minimal coming from the initial sequential code.

8.7. Extensions

This section discusses the different extensions that have been made to the POP-Java programming language to introduce new features. The work presented in this section comes from different projects that extend POP-Java and have been at least supervised by the author.

We start by discussing how to bring POP-Java to new distributed platforms such as the cloud (see 8.7.1) and YARN based infrastructures (8.7.2). We then finish the section with the concept of TrustedFriendComputing (8.7.3), a way to share computing resources through a network of “friends”.

8.7.1. Cloud integration

For the work describe in this section, the author had a supervisor role and did not create the described prototypes himself.

The POP model as well as POP-C++ and POP-Java have initially been developed with grid like infrastructures in mind. Yet over recent years, those types of environments have been largely replaced with cloud based approaches, either through private clouds or public ones like Amazon EC2. While bigger organizations still have grid like environments, most programmers do not have access to such infrastructure. With the introduction of the cloud, it became much easier for anybody to access large infrastructures.

As the reduction of complexity of distributed programming is the core goal of POP-Java, it makes sense to explore the possibilities to use the cloud to enable programmers to write distributed applications.

While fundamentally POP-Java works in a cloud based environment; after all they are virtual machines with IPs and the ability to run Java; we explored the possibility to make it easier for the programmer to exploit those infrastructures. Over the course of two Projects, by Xavier Barrelet [BKBW14] and Andrea Marcacci[MKW⁺13], we explored the requirements a cloud environment has and how POP-Java or the POP model in general need to be expanded to accommodate them. The cloud used to perform this exploration was the OpenStack²⁰ cloud, which is a cloud environment often used for private clouds. To access the OpenStack cloud, the JClouds library was used, which also gives access to other cloud types and is thus ideal for future expansions of the concept.

The main goal of the integration of OpenStack and cloud technology in general was to abstract the instantiation and destruction of cloud instances from the programmer. It is already possible to manually instantiate a number of cloud instances, configure them correctly to use POP-Java and run a POP-Java application. Through the integration we can create and destroy cloud instances when required to run new object, without having to worry to set up the required cloud instances beforehand.

Various additional configuration options have been added to the *@POPObjectDescription* annotation, allowing the programmer to specify the access parameters required to access the cloud environment. The creation and destruction of the virtual machines required to run the objects is abstracted from the programmer.

The prototypes were successfully finished and were able to run POP-Java applications on a private cloud installation. Nevertheless the code still has certain technical shortcomings which make further work necessary before merging the Cloud support into the main POP-Java version.

8.7.2. Hadoop cluster integration

Here we describe the work done to bring POP-Java onto Hadoop clusters. The author supervised to work and adapted the POP-Java languages in some aspects to make the project possible. The bulk of work was done by Mazzoleni Davide [MKPW16] during his Bachelor thesis.

Hadoop [Whi12] has become a hugely popular method to distribute various algorithms over a vast amount of computers. To use a Hadoop infrastructure, algorithms have to be transformed into a MapReduce [DG08] approach. While this is possible for some algorithm, others are more difficult to adapt to this computational paradigm.

With Hadoop becoming increasingly popular and thus Hadoop clusters becoming more available to programmers and researchers, we decided to explore the possibilities to run POP-Java applications on a Hadoop infrastructure. Details of this effort are documented in the Bachelor thesis *POP-DNA II, Parallel Object Programming in a Hadoop environment* by Mazzoleni Davide et. al [MKPW16]. As the main work environment we used the Hadoop cluster provided by DAPLAB²¹, which is a dedicated server infrastructure.

To achieve the goal of running POP-Java applications on a Hadoop cluster we did not use Hadoop directly, but YARN, which is a service that handles the resource allocation on a Hadoop cluster. Through YARN, resources can be reserved on a Hadoop cluster, which can be seen as similar to the way POP-Java and POP-C++ search for available machines to launch a certain object. Once a resource has been reserved, a container, which

²⁰<http://www.openstack.org/>

²¹<http://daplab.ch/>

is basically a lightweight sandbox environment on a Linux machine, is started and a user specified command is executed. We decided to run every new POP-Java object in one of the containers YARN provides.

By default POP-Java uses the POP-C++ JobManager, written in C++, to find and instantiate new objects on a POP cluster. For the integration of YARN, we decided to create an alternative JobManager, which performs this task by accessing the YARN API to reserve resources on the Hadoop cluster.

The JobManager was implemented in two versions. The first is a static version which allocates a user defined amount of YARN containers at the startup of the application. Those containers are then used during the lifetime of the application to instantiate new objects, just like on a traditional POP-Java cluster which is set up manually.

The second implementation uses a dynamic approach, only reserving as many YARN containers at a time as necessary. As the application is running, new containers are allocated on demand when new objects are created. Once the object in a container is no longer used, the whole container is destroyed to free the resources on the Hadoop cluster. Thanks to this functionality, effortless usage of a Hadoop infrastructure is possible for a developer, without having to worry about the details of how to handle the deployment of the different objects over the cluster.

On top of the integration of YARN as a resource allocator, new concepts have been introduced into POP-Java. Most notably the notion of object counters, which count the amount of active objects on a machine, or in this case a YARN container. This information is required to be able to destroy containers once they are no longer used, freeing the resources for other applications running on the same network. But this concept is also interesting outside of the YARN integration, as the JobManager can use it to determine on which machine to instantiate new objects on to more optimally distribute the workload over multiple machines.

The creation of the YARN prototype lead to various improvements of the POP-Java language, some of which have been integrated in the main branch which is available online. The integration of the complete YARN functionality into the official POP-Java version will require some more work, notably regarding the API which interacts with YARN. Currently the YARN API is directly accessed, which is not ideal due to its unstable nature and complexity. Projects like Twill ²² provide an abstracted API over the YARN service, providing a stable as well as easier API. The integration of YARN also noticeably increases the size of the POP-Java library, making it necessary to implement this feature in an optional way.

8.7.3. TrustedFriendComputing

The content of this chapter is partially based on the publication *FriendComputing : Organic application centric distributed computing* [WMK15] as well as the Bachelor thesis by Loïc Monney *Sharing computing power through a network of friends* [Mon14] which the author co-supervised. Further work on this concept has also been done through a bachelor semester project *FriendComputing-II* by Valentin Gazzola and a research project financed by the CTI (Swiss Commission for Technology and Innovation) for which the author is the project lead. The initial concept of TrustedFriendComputing was called FriendComputing, which was changed after more focus was put on the security aspects of the model. The author helped in an initial step to define the TrustedFriendComputing concept and its compatibility

²²<http://twill.apache.org/>

with the POP-Java vision. While for the two prototypes by Loïc Monney and Valentin Gazzola the author did not participate in the programming aspects, he does so for the CTI project during which the actual TrustedFriendComputing code is being created that is integrated into the publically released POP-Java version. The author also wrote the mentioned published paper [WMK15].

Introduction

POP-Java allows developers to easily create distributed parallel applications without having to worry about the details of distribution and interactions over the network. While making it easier to develop distributed applications, it is not sufficient for a wider adoption of distributed computing. Most organizations or individuals do not have access to a distributed environment such as a grid or a cluster. Yet, both scientific and commercial applications require increasing amounts of computational resources as well as storage space. Bioinformatics and NGS data analysis is certainly one example of such a field where this is the case, but others such as weather forecasting, multimedia processing or material simulations face similar challenges. Especially with the constantly decreasing sequencing costs in the NGS field as well as new developments with the miniaturizing of sequencers, the problem of computing resources has become an increasing issue. What before could only be done by big laboratories, which often already had the required computing infrastructure, is now increasingly done by smaller laboratories which do not have the same infrastructure.

This is not a new problem and over the years many technical approaches have been developed to assist in such situations. The recent arrival of cloud computing is the most important technology assisting smaller organizations which require larger computing infrastructures than they commonly have. Cloud computing enables them to quickly and relatively easily access an almost infinite amount of computing power, provided they are willing to pay the cloud providers for their services. In many domains, the usage of cloud computing is an acceptable solution which helps to decrease cost and allows for a quick adaptation to new computing infrastructure requirements. But not in all use-cases cloud computing is the ideal, or even acceptable, solution. When using cloud computing, all data and computation is offloaded to an outside location, not only outside the organization which initiates the cloud usage, but often even outside the country. When working with sensitive data, such as medical data, this is often not an option as many countries have laws which prohibit sending medical data over the internet or to another country.

But cloud computing is not the only solution when dealing with a limited computing infrastructure in a smaller organization. Over the years many grids have been created where multiple organizations pooled their resources together to create one common computing infrastructure. Compared to cloud computing, the organizations using that infrastructure can more easily control the data privacy and laws involved. Indeed, over the years many different grids, in a commercial and academic context, have been created. Some of the bigger and known examples are the Open Science Grid (OSG) [PPK⁺07] and the Worldwide LHC Computing Grid (WLCG) [BF07], which are traditional grids with multiple users being able to run a variety of applications on the grid. A different type of grid has also been created, with examples like Folding@Home [LSSP09] and BOINC [And04], which are application centric grids, generally only used by a single user for a specific analysis task.

While both of those approaches, cloud computing as well as grid computing, provide possible solutions to the problem of increasing amounts of data to be analyzed by increasingly small organizations, they both have drawbacks in terms of data privacy (mainly for the

cloud) and software development complexity (mainly when working with a grid). For example a small genetic laboratory, even if collaborating with other small genetic laboratories, often does not have the technical know how to setup such an infrastructure.

A common use-case is when multiple smaller organizations use the same software and that work on similar data and lack computing power during their peak resource usage. This is why we propose the TrustedFriendComputing model as an extension to POP-Java, with the goal to make it possible for application developers to develop distributed applications without the need for a dedicated distributed environment.

The notion of "Trusted Friend Computing" (TFC) has been coined at the Institute of Complex Systems (iCoSys) of HEIA-FR (HES-SO) in 2014. The main objective is to enable a community of users (called "friends") to securely share their IT resources without requiring that a central organization collects and stores all information. The community is built around the usage of a specific professional software application, in our case the graphical NGS data analysis pipeline.

To build such a community, the TFC model uses the notion of "confidence link", which is heavily inspired by the ViSaG security model [KCB14]. A confidence link is a bidirectional channel that allows two friends to communicate safely at any time. How the confidence links are established is not part of the definition of the model but is an hypothesis which defines the concept of community of friends. However, we can give as a possible implementation of a confidence link, a SSH channel between two computers whose friends have manually exchanged their public keys. The set of all friends together with all the confidence links form a connected graph, we can call a "trusted friend community", whose nodes are the friends and arcs are the confidence links. None of the friends in the community has a view of the overall infrastructure. Each friend only knows his direct friends, i.e. users with whom he has established a confidence link. Using this "trusted friends community", friends can securely share their IT resources for specific purposes related to the software application around which the community has been built. As an example of such a community, several Swiss hospitals located in different cantons, but offering molecular diagnostics on same diseases could easily safely and privately exchange genetic variant information and computing resources to reduce the delay and improve the quality of the diagnostics they offer. This project concerns the building of a system that allows to share IT resources in industrial environment.

In the next Section 8.7.3 we present different functionalities which have been implemented to test the concept of TrustedFriendComputing.

Results

To implement TrustedFriendComputing in POP-Java, a prototype has been developed which integrates the features needed to create an application which takes advantage of the TrustedFriendComputing model using POP-Java. Both POP-Java and POP-C++ were modified to achieve this. POP-C++ was modified to both remain compatible with the changes, but also because POP-Java does not currently have a JobManager but relies on the JobManager provided by POP-C++. In the POP model, the JobManager is responsible to find available resources in the network when creating new remote objects. To achieve the functionality of TrustedFriendComputing to automatically discover available computers in a network of friends, the JobManager is needed, which is why this prototype required both POP-Java and POP-C++.

The prototype successfully integrated the functionality to manage (create, delete) a network of friends as well as to invite to and join a network of friends. The programmer accesses this functionality through a new API integrated into the POP-Java language, which is optional to the usage of POP-Java and can be ignored if no `TrustedFriendComputing` functionality is used.

To connect to objects in a network of friends the programmer received two new annotations as well as a new way to choose the machine on which an object is executed. Using the already existing annotations, the programmer can specify the requirements of the machine he has, such as the amount of CPUs or RAM. With the new *Interest* annotation, the ID of the `TrustedFriendComputing` network can be specified. The second annotation, *Connect_to*, allows connect to an already existing object/application in the friend network.

When creating an object on a network of friends, there are more choices to be made than in a normal POP-Java network. In a normal POP-Java network, the object is on the first available machine which fits the criteria. But in a network of friends, there is a cost restriction (different host machines have different associated costs) as well as other potential restrictions, such as the geographic location of the remote host. Because of this, the programmer can implement a method (called *popChooseRemote*), in which he receives a list of potential candidates and can implement a custom choice of which remote object to use.

A prototype application was created that uses all those features and allows the user to perform multiple operations on a network of friend through a graphical user interface. For more details about this work, consult the previously referenced publication [WMK15] as well as the detailed report [Mon14].

After having proven the viability of the approach, the code of the prototype is now being adapted and integrated into the official version of POP-Java. Specifically for this purpose, a research project financed by the Swiss Commission for Technology and Innovation (CTI) was started at the end of 2016. The goal of the research project is to improve the concept of `TrustedFriendComputing` to make it possible to integrate it in a commercial application, making it possible to share resources between organizations in a secure way.

8.8. Usage

This section discusses how POP-Java was used during this thesis, in particular in the graphical NGS data analysis pipeline. Its main usage was in the distribution of sequence alignment. Future usage may come from `TrustedFriendComputing`, which will allow the creation of a distributed variant database (among other uses).

8.8.1. Distributed sequence alignment

In this section POP-Java is discussed in the context of distributed sequence alignment. The content of this section is based on the conference publication *POP-Java : Parallélisme et distribution orienté objet* [WKD14c] (POP-Java : Object oriented parallelism and distribution) published at the COMPAS 2014 conference. A special companion paper *Comment reproduire les résultats de l'article: "POP-Java: Parallélisme et distribution orienté objet"* [WKD⁺14e] was submitted to the REALIS 2014 satellite conference, which was dedicated to the reproduction of publications of the main conference. Through this paper, the results of the main paper [WKD14c] were independently reproduced by another research group.

While POP-Java has not been developed specifically for NGS data analysis, it helps programmers to develop distributed applications more easily. As NGS data analysis and sequence alignment in particular uses a lot of computing resources, it makes sense to speed up the calculations using POP-Java, which also helps to validate and improve POP-Java in the process. Indeed, doing distributed sequence alignment is a common approach, as seen in Section 7.1.3, but note that the method and code used for this chapter is different than the one described in Section 7.1.3. Both aligners are different, because the work presented and published in this section was done in 2014, whereas the previously presented aligner continued to evolve until the end of the thesis. In Section 8.8.1 we discuss the test setup used to test the sequence alignment distribution using POP-Java. The results of those tests are then presented in Section 8.8.1.

Methods

We developed a stand-alone sequence aligner based on the aligner presented in Section 7.1. Like the original aligner, this standalone aligner was developed using Java, targeting both single end and paired end sequences to be aligned. The resulting aligner was published as part of [WKD14c].

The algorithm is based on a streaming based approach, suited to both use multiple machines as well as multiple CPUs on every machine. Figure 8.6 shows the general architecture of the sequence aligner and how it is split into individual modules connected through streams.

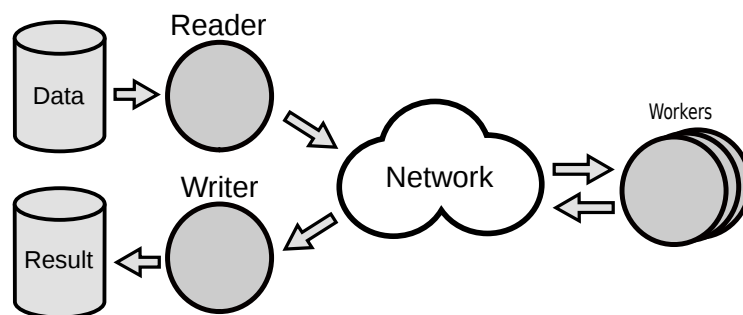


Fig. 8.6.: General architecture of the sequence alignment algorithm (adapted from [WKD14c])

The architecture of the aligner is ideal for distribution with POP-Java. Every module, the **Reader**, the **Writer** as well as the **Workers** are independent objects which can run in independent threads as well as on different machines. The interfaces between each of those objects have been clearly defined and can easily be distributed. For our implementation, every module represents a POP-Java object which can be run either locally or remotely. The amount of workers as well as their location is not limited, which allows the user to adapt the aligner to its needs and the currently available infrastructure. This also means that the **Reader** and **Writer** object do not have to be located on the same machine. This gives the user a big amount of flexibility. One example of this flexibility is to put the **Reader** object in one laboratory, the **Worker** objects in a dedicated grid and the **Writer** object on a machine in a second laboratory. This could be useful if for example one laboratory is responsible for the sequencing of a sample and a different laboratory performs the data analysis. This example is just one example of many which are made possible with an approach using stream based alignment as well as POP-Java to distribute sequence alignment.

To compare POP-Java with other implementations, we also used RMI (which is part of the standard Java library) to distribute the aligner.

The performance of the POP-Java implementation of the sequence aligner was tested on a grid with 6 machines, each having a quad core processor clocked at 3.4 GHz and 8 GB of RAM. The machines were connected with a 1Gb/s LAN network and for benchmark purposes hyperthreading has been deactivated. The dataset used consisted of 16 million DNA sequences which were artificially generated from the human chromosome 7. The 16 millions DNA were aligned only against the human chromosome 7.

In this setup, the Reader and Writer object were created on the same machine, with the up to 6 Worker objects being created on different machines. The Workers are programmed to use all cores of the local machine, which means that in our benchmarks we use between 4 and 24 CPU cores to align the sequencing data. The next Section 8.8.1 presents the results obtained during the tests on those machines.

Results

To evaluate the distributed sequence alignment using POP-Java we looked at two aspects. Code complexity as well as performance.

First is the code complexity required to create the distributed aligner, comparing the POP-Java and RMI implementation. The RMI as well as POP-Java implementation were done using 50 different classes. Out of those 50 classes, POP-Java required 3 of them to be POP-Java specific, and RMI 5 of them. The combined amount of source code lines of those implementation specific classes were 1200 lines of code for the RMI implementation and 300 lines for the POP-Java implementation. It is also to note that those 300 lines consisted of standard Java with only a few POP-Java specific annotations.

The next aspect we looked at was the performance when comparing both implementations. Figure 8.7 shows the total time required to align the data on 1 to 6 machines. Figure 8.8 shows the speedup calculated for both implementations compared to the optimal speedup.

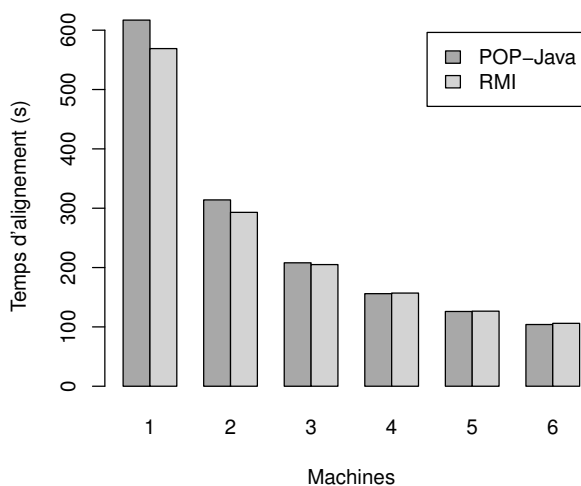


Fig. 8.7.: Alignment execution time on the machines, source [WKD14c]

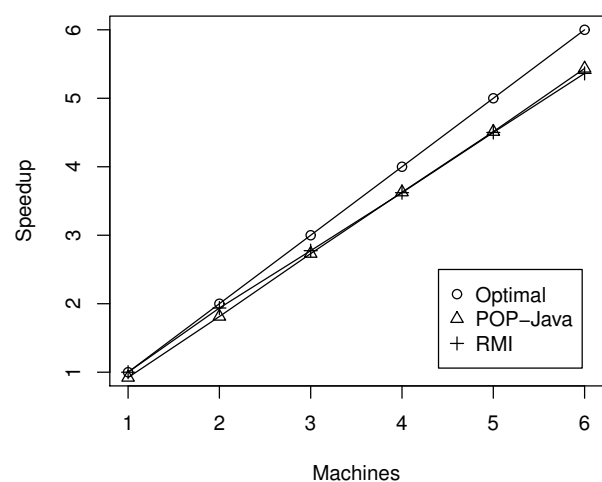


Fig. 8.8.: Alignment speedup over six machines, source [WKD14c]

The results show a nice speedup for the alignment using both technologies, RMI and POP-Java. Achieving the same performance with POP-Java as with a traditional Java

distribution technology was key. Those results show that distributing Java code using POP-Java achieves similar results as with other technologies, while at the same time greatly reducing the code complexity to achieve this task.

8.9. Limitations

POP-Java contains some limitations compared to pure Java which are looked at in this Section. While most limitations of POP-C++ also apply (see Section 3.5 in the POP-C++ manual), we discuss the major limitations which are specific to POP-Java and the Java language.

Parameter types

In Java, it is possible to send any type of object as a parameter to a method. In POP-Java, those parameters need to be serialized (and deserialized) as they have to be sent over the network. Thus, several restrictions apply to the types of parameters that can be used as method parameters of POP-Java methods. Those include that the objects need to either implement the interface `IPOPBase`, or be POP-Java objects themselves. While there are some exceptions, like the *String* object as well as all *Number* types (like *Double* or *Integer*), normal Java objects cannot be used as parameters for POP-Java methods. This also includes null objects, even if they are originally of a type that is supported by POP-Java.

Those limitations could partially be solved by writing custom serializers and deserializers for common types like *List*, as well as handling null parameters. But this approach could never handle all possible Java objects and there would be the problem of maintaining compatibility with POP-C++.

Parameter size

In Java, arrays have a certain maximum size which is defined by the size of a signed integer (2147483647), defining the amount of indexes which are possible in a given array. This results in different maximum sizes in terms of memory consumption depending on the type of array. A *byte* array for example would be at maximum 2.147 GB big, and an array of longs (which uses 8 bytes per value) would be at maximum 17.18 GB big. In pure Java, arrays with those maximum sizes can be passed as parameters to a method without a problem. In POP-Java however, those parameters are serialized into a *ByteBuffer* before they are sent to the remote method. With the serialization being done through a *ByteBuffer*, which follows the same rules as a *byte* array in terms of maximum size, the combined size of all parameters to a method cannot exceed 2.147 GB. This particular problem arrived when testing the matrix multiplication example in Section 8.6.1. To solve this problem, the serialization code of POP-Java needs to be rewritten to either use multiple *ByteBuffers* when serializing, or moving to a no copy approach. The later approach is much more interesting in terms of performance as well as memory usage, but requires a massive rewrite of not only POP-Java, but likely application developer for it as well. This is why currently this limitation is still present.

8.10. Discussion

We presented in this chapter a Java language extensions which implements the POP model, compatible with POP-C++. POP-Java allows developers to easily distribute ob-

jects over multiple machines to achieve distributed as well as parallel computing. Developers can download the open-source implementation of POP-Java from <https://github.com/pop-team/pop-java> and read more about it on the main website <http://gridgroup.hefr.ch/popjava/>.

We could show the viability of this approach of distributing calculations over a distributed system. This lead to the usage of POP-Java in multiple classes to teach students the concept of distributed computing.

Having laid a good groundwork, multiple projects are planned to make POP-Java more independent of POP-C++ (while staying compatible), introduce more features, like TrustedFriendComputing, and become even more user-friendly by adopting modern Java features.

Part III.
Applications

9. Graphical pipeline applications

In this chapter we discuss research that has been done with the help or in relation to the graphical pipeline presented in Chapter 6. The first section discusses research which has been done with direct author participation, which means that the author was listed as an author of the published work. This is followed by research which was done using the graphical pipeline, but where the author was not officially involved in the publications. The chapter is continued with an evaluation of GensearchNGS, through a survey made among users of the software. The goal of this survey was to study the impact of the graphical pipeline on the way research and diagnostics is done, especially regarding the complexity of the task.

9.1. Author participation

This section discusses different research in which the work of this thesis was used and the author was directly involved. Together with the following Section 9.2, it gives an overview of the real usages the software. The goal is to show that it has a practical impact on the genetic research community.

9.1.1. Deep intronic variants in the factor VIII gene

GensearchNGS has been used to analyze deep intronic variants in the factor VIII gene. The author actively participated in the research through the development of features needed to analyze the data in this study. The initial results were published as a poster presentation *Deep intronic variants in the factor VIII gene* at the "GfH Tagung" (Deutsche Gesellschaft für Humangenetik) 2015 [BWO⁺15a]. A more detailed and expanded analysis of those results was then published as a journal paper *Identification of deep intronic variants in 15 haemophilia A patients by next generation sequencing of the whole factor VIII gene* in *Thrombosis and Haemostasis* (2015) [BWO⁺15b]. This section is based on the content of both of those contributions.

Haemophilia is a hereditary bleeding disorder that affects the blood clotting. In affected individuals, the blood does not clot correctly and they continue bleeding. This can cause issues in various situations, for example when being cut. Haemophilia is linked to a gene defect in one of two genes, the factor VIII and factor IX genes. Haemophilia A (HA), the more common cause for haemophilia is caused by a defect of the factor VIII gene (F8). One in 5000 males [GBO⁺05] is affected by this disorder. Haemophilia B (HB) is less common and caused by a defect of the factor IX gene (F9). The rest of this section will discuss the more common haemophilia A variant, which was the focus of [BWO⁺15a] and [BWO⁺15b].

Correctly diagnosing an individual with a defect in one of the genes linked to HA is crucial to determine if its offspring is at risk of being affected by this particular disease. The methods currently used to diagnose haemophilia A work in a vast majority of cases.

Those methods screen the F8 gene mostly for casual mutations in and around the coding regions. But in about 2% of cases, the current methods fails and do not diagnose an existing haemophilia A disorder.

The haemophilia genes are located on the X chromosome and are recessive. As women carry 2 copies of the gene, they can be carrier or affected, whereas men with a defect gene are always affected. The gene, which has a total length of 187 kb, consists of 26 exons. Current methods search for various potentially causative sources in this gene. They currently search for inversions of intron 22 and intron 1, missense mutations and nonsense mutations in exons as well as deletions and insertions inside the exons or at splice sites. The detailed numbers about the different conditions can be found in [BWO⁺15b]. In about 2% of the cases, those standard sources of mutations do not reveal any defect. It is those 2% that have been investigated in more detail during this work.

In the study, 15 haemophilia A patients with a mild to moderate phenotype have been analyzed. The goal was to extend the current method of analyzing mainly the coding sequence of the gene and the inversions of intron 22 and 1, with the analysis of all intronic sequences. The intronic sequences were analyzed for SNPs and indels to see if the cause of the patient's haemophilia A could be identified in those additional targets not part of the standard methods.

GensearchNGS was already used in the laboratory of human genetics in Würzburg before this study. The detection and annotation of variants in the intronic regions of the factor VIII gene was already possible. But what was not yet possible was an automated and accessible way to analyze copy number variations (CNVs) in the NGS data, a crucial part of the analysis of those 15 patients. The CNV tool developed for GensearchNGS (see Section 6.7.6) was used to perform this analysis on top of the already existing tools in GensearchNGS to align the raw data (see Chapter 7.1), call variants (see Chapter 7.3) and analyze the variants. The detailed method to analyze the patients can be found in [BWO⁺15b], as well as the genetic discussion of the performed analysis.

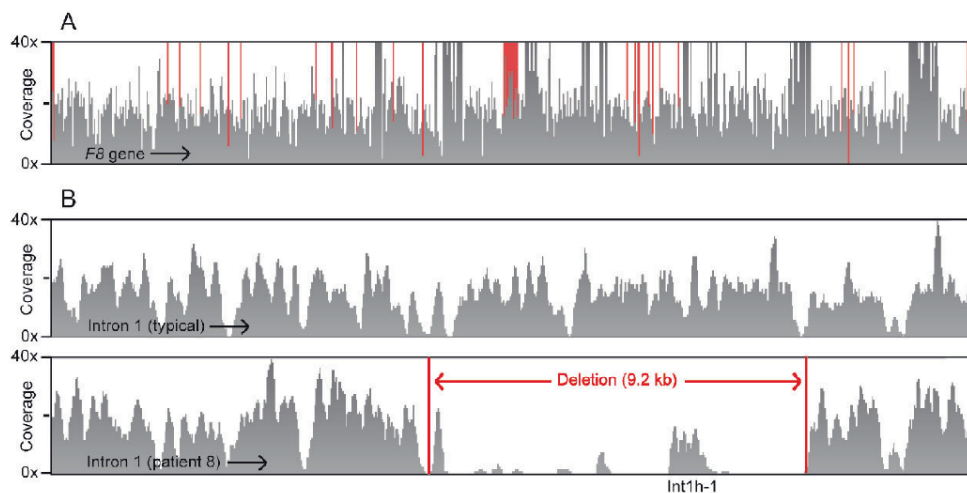


Fig. 9.1.: CNV analysis tool for patient 8 with a 9.2kb deletion in intron in. Figure taken from [BWO⁺15b]

In total, 23 deep intronic variants could be identified in the 15 analyzed patients. Every patient analyzed had at least one deep intronic variant that was identified as affecting the splicing of the factor VIII gene. One patient in particular showed a deletion of 9.2kb

in intron 1, as shown in figure 9.1, which shows the figure used in the published paper, showing the CNV analysis tool developed for that paper.

The success rate for the identification of deep intronic variations reached almost 100%, which is a very promising result for further studies trying to diagnose HA using NGS data. The paper led to new understandings about the nature of the causative variants leading to HA, allowing to diagnose even rare, previously non diagnosable causes of the disease.

9.1.2. Myofibrillar myopathies

This section gives an overview of some work that has been done with GensearchNGS related to myofibrillar myopathies (MFMs). The author participated in the poster *NGS panel for diagnostics of myofibrillar myopathies* [BRS⁺14]. The other articles discussed in this Chapter are *Novel recessive myotilin mutation causes severe myofibrillar myopathy* [SBR⁺14], *Unusual multisystemic involvement and a novel BAG3 mutation revealed by NGS screening in a large cohort of myofibrillar myopathies* [SSB⁺14] and the dissertation *Proteomische Einzelfaser-Analysen Myofibrillärer Myopathien mit Hilfe der Laser Dissektions-Mikroskopie* [Fel13]. They all used GensearchNGS in their research on myofibrillar myopathies.

Myofibrillar myopathies are disorders of the muscles which can cause weakness and muscle dysfunctions. Mainly skeletal muscles which affect body movement are affected, but also muscles in the heart can be affected ¹. Multiple genes are known to be associated with myofibrillar myopathies. Notably DES, CRYAB, MYOT, LDB3 (ZASP), FLNC, BAG3, FHL1, DNAJB6 and TTN [BRS⁺14] are associated with the disorder. The creation and analysis of a gene panel based on these genes for the use in diagnostics was proposed in [BRS⁺14], based on a data analysis with GensearchNGS. The previously mentioned works are now presented in chronological order in which they were published.

The published usage of GensearchNGS in relation to myofibrillar myopathies started in 2013 with a doctoral thesis by Sarah Feldkirchner. In her thesis she studied myofibrillar myopathies under the title *Proteomische Einzelfaser-Analysen Myofibrillärer Myopathien mit Hilfe der Laser Dissektions-Mikroskopie* [Fel13]. The focus of the thesis was not the analysis of NGS data, but data generated through laser capture microdissection (LSM) [EbBS⁺96] to further understand myofibrillar myopathies. Yet, certain parts of the validation were done using NGS data. This NGS data analysis was done at the University of Würzburg by the Department of Human Genetics, using GensearchNGS as the analysis software. One example of a result found through the analysis of the NGS data is a novel homozygous variant on the myotilin gene (MYOT).

During March 2014 the previously mentioned poster *NGS panel for diagnostics of myofibrillar myopathies* [BRS⁺14] was presented at the GfH (Deutsche Gesellschaft für Humangenetik) 2014. In the research presented in this poster, a new NGS panel for diagnostic screening of MFM patients was developed and evaluated. The panel consists of eight genes: DES, CRYAB, MYOT, LDB3 (ZASP), FLNC, BAG3, FHL1, DNAJB6 and TTN. 36 patients were sequenced and analyzed on all eight genes. 12 additional patients, for which preliminary Sanger studies showed no causative variants on seven of the eight genes, were sequenced on only the FLNC gene. The data analysis was done using GensearchNGS, which

¹ <http://ghr.nlm.nih.gov/condition/myofibrillar-myopathy>, 27.10.2015

included custom development of functionalities to enable the study. Five of the patients did show a causative variant on one of the sequenced genes. Two of those variants (in the genes BAG3 and TTN) were novel and not yet described in the literature. An additional nine variants with uncertain effects have been identified as well during this study. The results were deemed a success as a fast and practical way to perform diagnostic screening of MFM patients.

The two initial works were followed by two journal publications authored by (but not exclusively) the authors of the poster [BRS⁺14] as well as the doctoral thesis [Fel13]. Both publications were released at the same time and discussed the novel variants found in the previously discussed works in further detail.

The publication *Novel recessive myotilin mutation causes severe myofibrillar myopathy* [SBR⁺14] appeared in august 2014 in the journal Neurogenetics. It discusses in detail the homozygous variant found in the MYOT gene, previously described during the doctoral thesis of Sarah Feldkirchner [Fel13]. The NGS data analysis was again done using GensearchNGS.

The final publication in this series on myofibrillar myopathies that used GensearchNGS for the data analysis, was published in august of 2014 as well. This paper with the name *Unusual multisystemic involvement and a novel BAG3 mutation revealed by NGS screening in a large cohort of myofibrillar myopathies* [SSB⁺14] was published in the Orphanet Journal of Rare Diseases. In it, the authors go into further detail on the novel variant discovered on BAG3, previously presented in [BRS⁺14], as well as additional variants identified during the study. The publication lead to new recommendations regarding diagnostics of MFM patients using NGS. As with the initial poster [BRS⁺14], GensearchNGS was used for the data analysis.

The usage of GensearchNGS in multiple publications discussing myofibrillar myopathies lead to a continuous improvement of the software, based on the feedback of its users.

9.1.3. Transcriptomics

Adding RNAseq data analysis capabilities into the graphical pipeline allowed to participate in research in the domain of transcriptomics. Here we present the works done in the context of transcriptomics during this thesis. This section is based on the content published in *Non-coding RNAs in lung cancer: Potential as non-invasive diagnostic tools and bioinformatics analysis approaches* Submitted Oct. 2016 [KWS⁺ss].

We studied the role of non-coding RNAs in cancer and in particular lung cancer, is the most common cause of cancer related deaths. This is caused by the late diagnosis which limited the number of treatments available for this type of cancer. This is why a better understanding of lung cancer as well as new biomarkers which can be used for earlier detection are crucial. As described in recent studies, micro RNAs and long non-coding RNAs play an important role in the development of lung cancers. This makes them ideal study targets for both diagnostics as well as treatment

In the published paper we explore the current state of the art of the field and the possibilities on how to analyze lung cancer through RNA sequencing. A particular focus is put on the fact that in contrast to DNaseq, which often analyses particular genes and the proteins they produce, the function and interaction of micro RNAs and long non-coding RNAs is

not well documented. This is why a variety of databases as well as prediction tools are presented that help researchers to better understand their samples. We also explored how the process of RNAseq data in this context can be automated to make it more accessible and ultimately usable in the clinic. Many of the features implemented in the graphical pipeline came as a direct result of this paper, putting into practice some of the recommendations made in the publication. The graphical pipeline also served as a model to show what is possible in terms of integrated graphical NGS analysis pipelines.

9.1.4. Epigenetics

The addition of methylation analysis support in GensearchNGS made it possible for geneticists to analyze bi-sulfite sequencing data. This work led to two works in the field of epigenetics with author participation. A workshop presentation *The unmethylated allele of oppositely imprinted (i.e. MEST and MEG3) genes is highly susceptible to epimutations during early development and may contribute an additional layer of complexity to phenotypic variation* as well as a poster *Deep bisulfite sequencing for quantification of constitutive epimutations in tumor suppressor genes* [BAW⁺16] were presented during the “Jahrestagung der Deutschen Gesellschaft für Humangenetik” 2016. The amount of user-friendly tools to analyze bisulfite methylated NGS data is still very limited as of today. This is one of the major roadblocks for a broader adoption of this method in the case of the human genetics department of Würzburg. On the basis of the DNaseq analysis capabilities, GensearchNGS has been extended to analyze and visualize bisulfite sequencing data. Those enhancements, detailed in Chapters 7.6.4 and 7.5, made it possible to perform the analysis needed for those two works.

Differentially methylated genes coming from the paternal or maternal side have an important role in early growth and metabolism. Being able to analyze and visualize this data is important to understand the differences in methylation for the paternal and maternal copy of a gene. As with all NGS technologies, it is not possible to know from which chromosome (paternal or maternal) an individual sequence originates from. To solve this problem, for this particular study, heterozygous individuals have been selected, all having a particular variant only present in one of the parents on the genes being analysed. Visualizing as well as analyzing the methylation of a specific region of bisulfite NGS data, based on the presence or absence of a certain variant was one of the major points to solve for this analysis. This made it possible to separate the data of the paternal and maternal copy of the gene in a user-friendly manner, making it possible to look at the data.

The work done to allow the data analysis for the analysis of differentially methylated genes in maternal and paternal genes was also used to develop a new method which aims for screening the methylation of tumor suppressor genes to determine cancer risk. The underlying motivation for this new method is the fact that the most commonly used test to determine hereditary breast and ovarian cancer risk, which is based on the two genes BRCA1 and BRCA2, detect less than 25% of those two types of cancer. It has been shown [HPL⁺12] that epigenetic mutations in tumor suppressor genes like BRCA1 or RAD51C can explain a certain percentage of the previously undetected cancers. To be able to rapidly determine the cancer risk associated with epimutations in tumor suppressor genes, an analysis protocol based on deep bisulfite sequencing has been implemented and tested. In a first pilot study 96 female patients have been analyzed on 10 tumor suppressor genes (BRCA1, BRCA2,

ATM, PTEN, RAD51C, MLH1, TP53, MLH1, ESR1, and RB1). The patients came from two different groups. The first consists of 48 samples which have been taken from patients with early onset breast cancer. The second group served as a control group with 48 healthy female patients in the same age category. Based on those two groups the goal was to develop biomarkers for cancer risk prediction. GensearchNGS had to be adapted, in a similar way as for the analysis of differentially methylated maternal and paternal genes, to allow that kind of analysis. Especially important was the integration of the ability to visualize and compare the methylation between many samples for specific genes. This initial study did not yet lead to the discovery of new biomarkers which can be used in the clinic. But interesting epimutation patterns in certain individuals have been observed and give hope that the analysis of a larger cohort could lead to the desired results.

9.2. Indirect participation

This section gives an overview of the research made using our graphical pipeline. We do not intend to make a detailed analysis of every publication, but to give an overview of what has been made possible through the usage of our graphical pipeline, GensearchNGS.

In the publication *A Novel de novo Mutation in CEACAM16 Associated with Postlingual Hearing Impairment* by Hofrichter et al. [HNG⁺15], GensearchNGS has been used to detect the variants present in a trio, sequenced with the TruSight One NGS panel. The variants have also been annotated, visualized and filtered through GensearchNGS. In the results, which have been published in the journal *Molecular Syndromology* during the august 2015, the authors showed a de novo mutation in the CEACAM16 gene. This de novo mutation could be linked to hearing impairment, which further expands the knowledge about the CEACAM16 gene and enables new research directions when diagnosing hearing impairments.

The publication *Diagnostic approach for FSHD revisited: SMCHD1 mutations cause FSHD2 and act as modifiers of disease severity in FSHD1* by Larsen et al. [LRE⁺15] was published in the *European Journal of Human Genetics* during November 2014. In it, the diagnostics approach to Facioscapulohumeral muscular dystrophy (FSHD) is revisited, as the study shows the implication of the SMCHD1 gene in FSHD. 55 FSHD type 1 negative and 40 FSHD type 1 positive patients have been sequenced using the 454 GS Junior platform. GensearchNGS was used to perform the complete data analysis. This includes sequence alignment, variant calling/analysis and visualization.

Also discussing muscular dystrophies, the publication *Identification of variants in MBNL1 in patients with a myotonic dystrophy-like phenotype* by Larsen et al. [LKS⁺16], published in the *European Journal of Human Genetics* during mai 2016, focused on myotonic dystrophies. 138 patients were sequenced using a 454 GS Junior sequencer by Roche, on the MBNL1 and CELF1 genes. Additionally, for 90 of those patients the DMPK and CNBP genes were sequenced. The work done mainly helped to further support the important role the MBNL1 gene has in myotonic dystrophies. The analysis including alignment and variant calling was done using GensearchNGS.

The previous two publications were part of the thesis *Zur genetischen Heterogenität der Muskeldystrophien : alternative genetische Ursachen der Myotonen Dystrophie und FSHD*

by Mirjam Larsen [Lar15]. The thesis was published during the December of 2015. The first part of the thesis focused on the work published in a previous publication where FSHD was studied. The second part of the thesis focused on muscular dystrophy (MD) type 1 and 2. Both parts of the thesis used GensarchNGS as one of the tools for NGS data analysis.

In April 2015, the publication *Nail-Patella Syndrome: clinical and molecular data in 55 families raising the hypothesis of a genetic heterogeneity* by Ghoumid et al. [GPHE⁺15] was published in the *European Journal of Human Genetics*. In this publication, the Nail-Patella Syndrome was studied, a rare autosomal dominant genetic disorder which leads to poorly developed nails and skeletal anomalies. The study consisted of 55 patients and 39 of their relatives. The samples were analyzed with both MLPA and Array-based comparative genomic hybridization (CGH). Both of those techniques are used to detect large copy number variations. If both of those tests were negative, the samples were sequenced with the 454 GS Junior platform and analyzed with GensearchNGS. This has been done for 5 out of 55 patients, as the other techniques did not reveal any relevant findings.

In the publication *Novel form of X-linked nonsyndromic hearing loss with cochlear malformation caused by a mutation in the type IV collagen gene COL4A6* by Rost et al. [RBN⁺14], COL4A6 has been identified as the fourth gene linked to X-linked nonsyndromic hearing loss. The publication was published in May 2013 in the *European Journal of Human Genetics*. Two affected males and one carrier female subject were sequenced using the HiSeq2000 sequencer from Illumina. Another 94 male and two female patients have been sequenced using the Genome Sequencer 454 FLX System by Roche. The samples from both sources were analyzed with GensearchNGS, with the alignment having been done using the BWA aligner.

Another usage of GensearchNGS can be found in *Next-generation DNA sequencing of a Swedish malignant hyperthermia cohort* by Broman et al. [BKB⁺15]. This publication was published in *Clinical Genetics* in October 2014 and explored the usage of next generation sequencing in diagnostics for malignant hyperthermia. Malignant hyperthermia is a rare disorder which can lead to the death of a patient when undergoing a narcosis. Multiple causes for the disorder exist, one of which is a genetic defect. The study analyses a panel of 64 genes, with 5 patients being sequenced using the Illumina HiSeq2000 sequencer. The data was analyzed with GensearchNGS, including the sequence alignment, variant calling and visualization.

In a different research direction, although still in human diagnostics, GensearchNGS was used in *ALS and MMN mimics in patients with BSCL2 mutations: the expanding clinical spectrum of SPG17 hereditary spastic paraplegia* by Musacchio et al. [MZÜ⁺16]. Here, GensearchNGS was used to analyse NGS data coming from a TruSight Exome sequencing to detect de novo mutations in a patient in a trio analysis. The publication led to new insights into the hereditary spastic paraplegia disease.

GensearchNGS was also used for research on male infertility. *The human RHOX gene cluster: target genes and functional analysis of gene variants in infertile men* by Borgmann et al. [BTD⁺16] looks at the role of RHOX gene cluster on male infertility. GensearchNGS was used for the data analysis of the NGS data used in this research project.

One of the first uses of GensearchNGS was the analysis of hereditary breast and ovarian cancer (HBOC). Multiple laboratories perform this type of analysis using GensearchNGS

nowadays, with the laboratory of human genetics at the University of Würzburg being the one we collaborate most closely. Initially only the two most known genes BRCA1 and BRCA2 were analysed, a list which was expanded upon later to include also a larger list of genes associated with HBOC.

This led to an initial poster which analyzed 300 patients using GensearchNGS, *Hereditary Breast/Ovarian Cancer: A systematic screening of DNA repair genes in 300 consecutive patients* [AG15] at the GfH Jahrestagung, 2015. This work was later expanded to an even larger gene panel and 500 patients, presented in the poster *Hereditary breast/ovarian cancer: a systematic screening of 94 cancer associated genes in 500 consecutive index cases* [AG16] at the ESHG 2016. Detailed statistics of the distribution of variants related to HBOC over the different genes of the gene panel could be presented. In a similar light, the poster *Analysis of 37 / 65 muscle genes in 300 patients with neuromuscular diseases* by Pluta et al. [NP16] presented at the ESHG 2016 analyzed 300 patients. The integrated variant database in the graphical pipeline was used to create overview statistics to increase the understanding of neuromuscular diseases. Those works are also important as they serve as a validation of the methods used in GensearchNGS.

9.3. User survey

This section analyses a user survey which was created to better understand the impact our graphical NGS data analysis pipeline has on the life of researchers and geneticists in general. The survey was filled out by various users of the different genetic laboratories that use the software during summer 2016. The focus of the survey was to determine if the software does indeed help geneticists to more easily work with their data, as well as make them more independent from bio-informaticians for day to day tasks. We look more in-depth at the demographic as well as the goals of the survey before discussing its results, which also included feedback about the software coming from outside this particular survey.

9.3.1. Introduction

Determining the impact of a software on the daily life of its users and especially determining if it meets the goals that were defined when it was initially created is difficult to test. This is why we decided to make a user survey. As the graphical pipeline developed during this thesis is actively used in several European genetic laboratories, we asked them to fill out the survey.

Thus we can define the target demographic users of our graphical pipeline, which used it at least once but ideally regularly. Most of them are geneticists or bio-informaticians.

The survey questions had four major goals. The first was to better understand the background of the users which use the software. This includes their scientific background and if they use the software for research or diagnostics. The second goal was to determine how often and for what purpose the users use the software. This allows us to determine if the software is actually used on a regular basis. The third, and probably most important goal, was to determine if the software meets its initial goals, which is to lower the complexity of the data analysis so that geneticists can work more independently in many situations. The last goal was to have a general feedback about the software and how it could be improved in the future.

The next section presents the results of the survey.

9.3.2. Results

We surveyed 16 users of the graphical pipeline during the summer of 2016 to better understand the user-base as well as the impact of the software. The laboratories that participated in the survey are: The laboratory of human genetics at the University of Würzburg, the institute of human genetics at the Universitätsklinikum Münster, the University of Lille and others we don't name due to the anonymous nature of the survey. The users were presented with 24 questions from four categories, organized according to the four goals presented in the previous section.

- *User background* - Questions about the background of the user
- *Software usage* - Determine how often and for what purpose the software is used
- *Usability* - How good is the usability of the software and does it help the users
- *Feedback* - General feedback on how to improve the software in the future

We analyse the results in the following sections. The detailed results can be found in Appendix D.1.

User background

Out of the 16 users, 9 self identified as geneticists, 3 as biologists and the rest with one genetic counselor, one technical assistant and one PhD student. On a scale of 1 (beginner) to 6 (expert), the majority (10) evaluated their computer skill as slightly above average at 4. When asked to describe the type of work they are doing, the most common keywords were cancer related (8 times) and concerning rare diseases (4 times). The details of this questions can be found in Appendix D.12.

Software usage

Most users, 62.5%, have been using GensearchNGS for 2 or more years and 18.8% used it for one year. The rest, 6.3%, used it for 6 months and 12.5% for one month (Figure 9.2). Not only have most users been using GensearchNGS for a considerable amount of time, but they also use it regularly as seen in Figure 9.3. 43.8% use it daily, 31.1% weekly and the rest use it either once a month or do not use it regularly.

The users not only used it for some time, but also analyze a considerable amount of samples using the pipeline. On average they analyze about 230 samples per year, with the highest reported number being 1000 and the lowest 12 (see Appendix D.5 for details).

This is done mostly in a research and diagnostics context. The users could give multiple answers to the question of the context usage, with 81.3% using it in research and 75% in diagnostics.

The type of data being analysed is mostly DNaseq data (87.5%), with only 12.5% doing bisulfite sequencing analysis.

As seen in Figure 9.4, the amount of people analyzing somatic (37.5%), germline (37.5%) or somatic and germline (25%) variants is balanced.

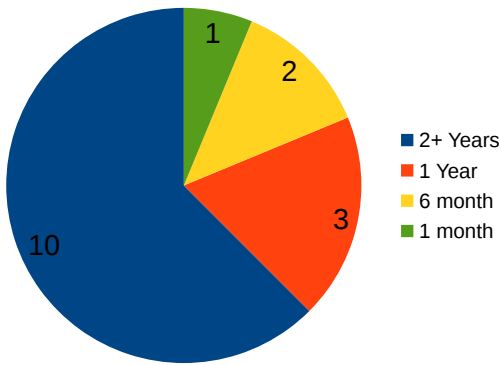


Fig. 9.2.: How long have you been using GensearchNGS?

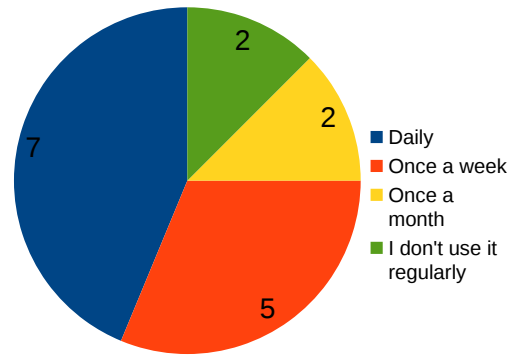


Fig. 9.3.: How often do you use GensearchNGS?

In Figure 9.5 we can observe that most users (62.5%) import aligned data (BAM files) directly into the software. An additional 37.5% even went one step further and imported the variant files (VCF) in addition to the aligned data into the software. Only 25% of the users actually go through the full graphical pipeline, going from raw data to the final variant report. The users had the possibility to choose multiple options, which is why the total is higher than 100% (see Appendix D.9).

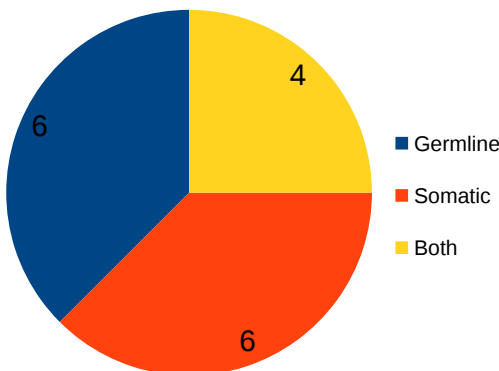


Fig. 9.4.: Do you analyze germline or somatic mutations?

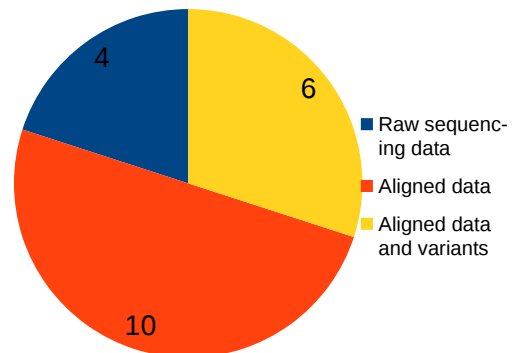


Fig. 9.5.: What format do you use when importing raw data?

To complement the data analysis, the users use several other software tools, like Exomiser[RKO⁺14], the torrent suite variant caller and MiSeq reporter. The complete list can be found in Appendix D.11.

Usability

Several questions about the usability of the software were asked. The users were asked to rate several statements with a value going from 1 to 6, with 1 being totally disagreeing and 6 totally agreeing. The questions were optional, so that users could only answer the ones that directly concerned them.

The intent of the questions was to determine if the goal of reducing the complexity of OMICs data analysis could be reduced by using the software we developed.

The first question aimed to determine if the overall goal, which was to lower the complexity of OMICs data analysis, was achieved. Figure 9.6 shows that a big majority agreed that the developed software achieved this goal. In Figure 9.7 we see the answers to the question of the software allows to save time compared to its alternatives. Again a majority of agreed that the software saves them time, although 4 people did not agree with that statement. Those four people cited NextSeq, Amplifyer, SnpEff and custom solutions as the other software they are using, an information which we can use to determine future features to prioritize.

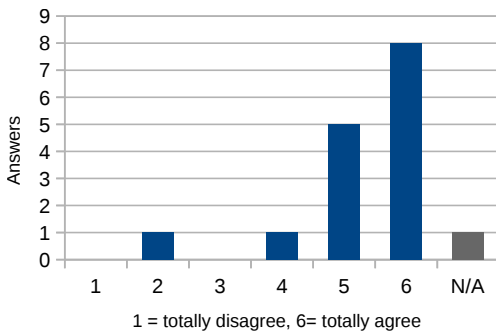


Fig. 9.6.: It facilitates the analysis of complex data sets (DNaseq, RNAseq, DNA methylation)

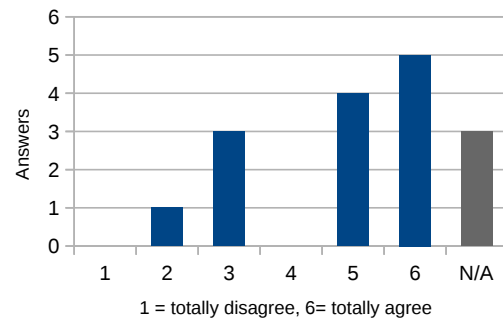


Fig. 9.7.: It allows to save time compared to other software

Again most users agreed that the software is easier to use than its alternatives (Figure 9.8), and that it also reduces the dependency of biologists and scientists on bioinformaticians (Figure 9.9).

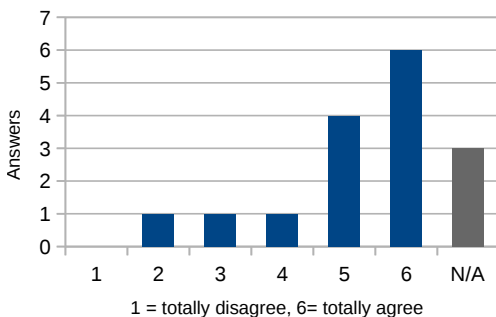


Fig. 9.8.: It is more user-friendly than its alternatives

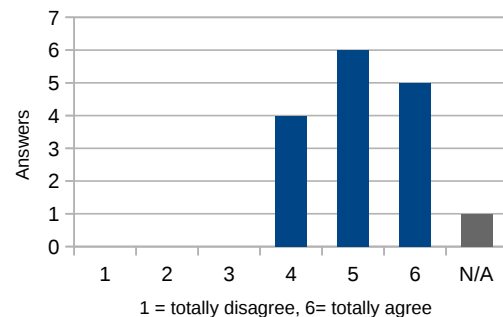


Fig. 9.9.: It reduces biologists/scientists dependence on bioinformaticians

Also interesting to note is that most respondents use GensearchNGS as their main NGS

data analysis software (Figure 9.10). The question about performance on older slow computers was only answered by 11 out of 16 respondents (presumably because the others do not have slow computers). Among those 11 respondents, they rated the performance positively on slow computers (Figure 9.11).

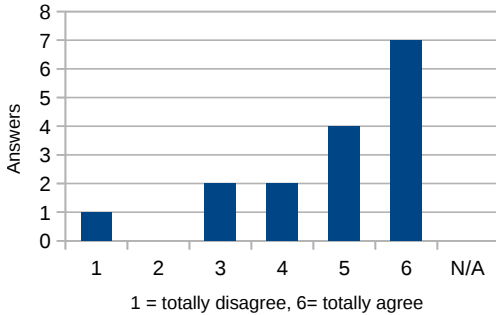


Fig. 9.10.: GensearchNGS is my main NGS data analysis software

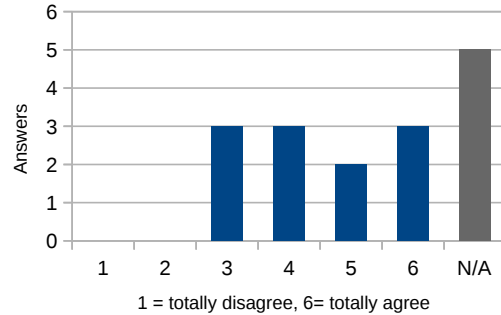


Fig. 9.11.: It works reasonably fast on older/slower computers

For the two last questions, the users were asked if they feel that they can perform diagnostics (Figure 9.12) as well as research (Figure 9.13) more efficiently. Both were answered with a clear yes.

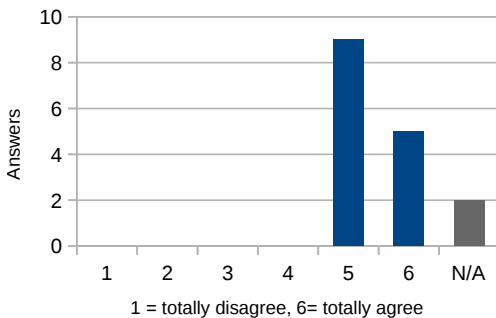


Fig. 9.12.: It helps me to do diagnostics more efficiently

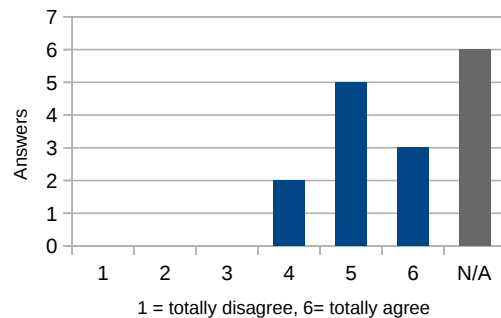


Fig. 9.13.: It helps me to answer research questions more efficiently

Feedback

The last questions of the survey were general feedback questions in which the user could give free text feedback.

The answers for the “most useful functionality” shows the diverse use-cases of the users. As expected, most use the software to analyse variants, which is why features around variant analysis are the ones to be mentioned most. But features from other domains, like CNV analysis and the genome browser are also mentioned. This shows that the users are happy with many different aspects of the graphical pipeline and don’t use it for a single functionality. A complete list of the mentioned features can be found in Appendix D.21.

The next question asked about missing and incomplete functionality. Again the answers were quite diverse, but with the CNV analysis being highlighted more than once. Again the complete list can be found in Appendix D.22.

The last question was a general feedback form, which was mostly used by the respondents to praise the software and how useful it is for them. One positive quote from the feedback is “best NGS analysis software I ever used”, which is encouraging and shows how much the users appreciate the pipeline. The complete feedback list can be found in Appendix D.23.

9.3.3. Summary

Through the survey we could evaluate the type of users the graphical pipeline attracts. Unsurprisingly, the users pretty much fall into the target audience of geneticists doing DNaseq in a diagnostics and research environment. Not very surprising but still notable is the low amounts of users of both RNAseq and bisulfite sequencing. We attribute this to the fact that RNAseq and especially the bisulfite sequencing, while being hot research topics, are not yet common in diagnostics laboratories as DNaseq.

Another interesting fact we learned through is that only one quarter of the users actually go from raw data to the final analysis results. We don't know if this is because the laboratories already have a dedicated infrastructure to align the data or if the computers used to run the graphical pipeline are not well equipped enough to perform the analysis.

An encouraging fact is that the users that responded use the pipeline regularly and analyze on average 230 samples per year, which is a significant amount of data. In terms of the goals we were set to achieve, which were to render NGS data analysis more accessible, the users seem to agree that this is the case. They also answered positively that the underlying goals of this point, which is to make biologists more independent of bio-informaticians and to make both diagnostics and research more efficient.

For the second goal, which was to resolve the performance problems which accompany NGS data analysis we asked the question about GensearchNGS being reasonably fast on older and slower computers. While only 11 out of the 16 users answered this question, the overall feedback was positive on this account as well. Considering that most users do not align the data inside the graphical pipeline (which is the most time consuming analysis step), efforts like the improved variant calling and fast variant filtering may contribute to this result.

9.4. Discussion

The stated main goal of this thesis is to create a more accessible NGS data analysis approach as well as to reduce the performance problems associated with the data analysis. The graphical NGS data analysis pipeline created in that process has been used in various research projects, as discussed in Sections 9.1 and 9.2. The variety of works published using the graphical pipeline show how the chosen approach adapts to different types of analyses. The various usages in published works also serve as further validation for the validity of the analysis results given by the graphical pipeline.

Going further than only looking at the actual usages of the graphical pipeline, we wanted to evaluate if the goals set out to achieve were actually met. As assessing questions like the accessibility is hard to do, we decided to perform a user survey to see the impact the software has on its users.

As discussed in Section 9.3.3, the survey shows that we did indeed achieve those goals, which together with the list of published results using the pipeline, are encouraging results.

10. Conclusion & Future works

Over the course of our research, we tackled the problem of OMICS data analysis complexity through three approaches. Those three approaches address the issues of the reduction of the technical skill requirements, the computing infrastructure requirements and to ease the development of distributed applications in order to speed up the data analysis. We will now discuss what has been done to address those three issues, our findings and what should be done in the future to further improve them.

Reducing the technical skill requirements to perform OMICS data analysis brings us to one of the major contributions of this thesis. Serving as a center piece, we developed a fully featured graphical NGS data analysis pipeline which is actively used in the research and diagnostics community (Chapter 6). The main goal of this graphical pipeline was to reduce the complexity of OMICS data analysis, which is caused by the existing multitude of nonintuitive data analysis tools, that are often not compatible with each other. This was achieved by creating a user-friendly interface, integrating the common tools required in a diagnostics environment as well as the direct or indirect integration of external resources and databases. The graphical pipeline has been used extensively in a diagnostics environment and allows biologists to more easily perform their routine work, especially at the human genetics department of the University of Würzburg. This led to several published works by other authors which have been realized thanks to our graphical pipeline (Chapter 9). They cover various domains of genetics, sometimes leading to new methods. The usage of our tool in research, which in large parts came from its usage in daily diagnostics, is a good indication that we achieved our goal. Nevertheless, to better assess the impact of our graphical pipeline we made a user survey (Chapter 9.3). It showed that the users agreed with the assessment that the graphical pipeline lowered the complexity of OMICS data analysis, both in a diagnostics and research environment. Through our work we saw the importance of easy to use interfaces which guide the users through the analysis, while at the same time not putting too many restrictions on the type of analysis that can be performed. Even if it might not be possible to create an intuitive solution for every type of OMICS analysis task, we showed that a specialized software solution for this problem of data analysis can help people to work more efficiently. This work focused on making the data analysis easier for diagnostics labs. We think that at medium-term, through the combination of more user-friendly analysis tools and cheaper sequencing, it will become possible for everybody to analyse their genome at home.

On top of having an intuitive interface which allows more people to perform OMICS data analysis it is crucial to make the required infrastructure accessible. To achieve this, we explored the possibilities to lower the complexity of OMICS analysis by reducing the required computing infrastructure. We tackled this issue by first optimizing existing methods and secondly by using parallel and distributed computing. We published the results of our optimized variant calling, which produces the same results as an existing method but up to 18 times faster, shows how much potential there is in existing methods (Chapter 7.3).

This success, also outside of the variant calling analysis, is also confirmed by the results of the user survey.

We not only investigated improvements to existing methods by writing more optimized code, but also how to use distributed computing to speed up the data analysis, in particular sequence alignment (Chapter 7.1). While for this task, performance was not our main goal (as existing sequence aligners are already well optimized), we focused on allowing the distributed algorithm to optimally use the available resources. We showed that it is possible to use distributed alignment in various types of environments, making it possible to adapt to the specific needs and restrictions of specific users. This is in contrast with many existing methods which rely on either pre-configured clusters or clouds. Our approach is not only able to use those infrastructures, but also dynamically scale over multiple computers in the same network, and combine the different types of infrastructure.

To lower the complexity of distributed application development, we designed a Java language extension, called POP-Java (Chapter 8). We brought the POP model to the Java programming language, allowing programmers to transparently distribute the workload over multiple computers by distributing objects. We showed that the performance of the approach is similar to existing methods, while reducing the amount of code needed to develop the same application. Due to its properties, the programming language has also been used in distributed programming classes at the master level.

The concept of FriendComputing, which was introduced in the thesis, resulted in the financing of an ongoing research project on the subject. This research project leads the way to the future of the POP-Java language and hopefully distributed programming in general. Its goal is to provide programmers as well as end users with simpler tools to distribute calculations on decentralized and dynamic networks, all while providing the best possible security. The security aspect is especially important in the context of clinical diagnostics, where sensible human DNA is handled. As shown in [GMG⁺13], individuals can be identified by their anonymized sequencing data, raising concerns over the usage of remote computing resources like the cloud for NGS data analysis. This problematic will remain a major research in terms of clinical diagnostics in the future, as distributed computing is one of the ways to handle the ever increasing amount of data in that domain. Being able to handle them securely in a distributed environment will become critical. Even if approaches like homomorphic encryption [Gen09] exist, which perform calculations on encrypted data, the performance of those systems is still far away from being practical.

The mentioned improvements are part of the future work, which are various. They include better data visualization, support for more types of data analyses and improvements of the existing tools that perform those analyses. Due to the rapidly changing technological landscape of genetic data analysis, there is indeed no shortage of improvements which can be explored. To name one example, the increasingly popular long read sequencing technologies of the latest sequencing generation, like the MinION, create new challenges and opportunities for bio-informatics analysis tools, like the ones we developed in this work. The long read size and high error rates of those new technologies, require new approaches to handle that type of data and to extract the relevant information.

An even bigger change will come from the adoption of graph based reference genomes [NHG⁺17], which represents the reference not as a one dimensional string, but as a graph which can represent the vast variety of genomes found in a population. But such a change will be incompatible with most existing tools, requiring major, but useful changes. Due to

the increased complexity of the data analysis that comes from graph based references, it will be even more important that graphical pipelines like the one presented in this thesis, guide the user through the analysis process.

On a higher level, increasing the synergies between the different OMICs sources, to improve the analysis quality, is also a focus for future works. While NGS data analysis tools, like ours, already start to support multiple OMICs data types, they do not yet allow the user to combine them in an intuitive way to gain more insight into a sample. To analyse a sample using multiple OMICs data-sources is a costly and complex task, for which intuitive user interfaces will be required to extract the relevant information from a sample.

All those future works have the potential to increase the computing infrastructure requirements, which makes our work on distributed computing even more important. Being able to use distributed computing to lower the infrastructure requirements is also helpful in regards to the ability to analyse genomes at home. Offloading the computation heavy calculations to the cloud, privacy issues aside, will help to overcome the computing infrastructure limitations found at home.

The leitmotif of this thesis was to make people's lives easier, allowing them to work more efficiently. We showed that the presented work achieves this goal in the fields of bioinformatics and distributed computing.

11. Publications

This chapter lists the works published during this thesis. Section 11.1 contains the list of publications both in journals, at conferences and workshops. In Section 11.2 all poster presentations are listed.

11.1. Publications

We separated the publications in three groups. In the first group we have the publications released in peer-reviewed journals. They are followed by peer-reviewed conference publications. Lastly we have publications released during conferences and workshops with a less strict peer reviewing process (on invite etc.).

11.1.1. Journal papers

Meik Kunz, **Beat Wolf**, Harald Schulze, David Atlan, Thorsten Walles, Heike Walles and Thomas Dandekar, *Non-Coding RNAs in Lung Cancer: Contribution of Bioinformatics Analysis to the Development of Non-Invasive Diagnostic Tools* [Review], Genes, Dec. 2016, 24 pages

Beat Wolf, Pierre Kuonen, Thomas Dandekar, David Atlan, *DNaseq workflow in a diagnostic context, and an example of a user friendly implementation*, BioMed Research International, Volume 2015 (2015), Article ID 403497, 11 pages

J. Elisa Bach, **Beat Wolf**, Johannes Oldenburg, Clemens R. Müller, Simone Rost, *Identification of deep intronic variants in 15 haemophilia A patients by Next Generation Sequencing of the whole factor VIII gene*, Thrombosis and Haemostasis, 2015: 114/4 (Oct) pp. 657-867

11.1.2. Conference proceedings

Beat Wolf, Pierre Kuonen, Thomas Dandekar, *GNATY: Optimized NGS variant calling and coverage analysis*, 4th International Work-Conference on Bioinformatics and Biomedical Engineering (IWBBIO 2016)

Beat Wolf, Pierre Kuonen, Thomas Dandekar, *POP-Java : Parallélisme et distribution orienté objet*, Compas2014, Conférence d'informatique en Parallélisme, Architecture et Système

Beat Wolf, Pierre Kuonen, Thomas Dandekar, *Comment reproduire les résultats de l'article: "POP-Java: Parallélisme et distribution orienté objet"*, Realis 2014: Reproductibilité expérimentale pour l'informatique en parallélisme, architecture et système

Beat Wolf, Pierre Kuonen, *A novel approach for heuristic pairwise DNA sequence alignment*, BIOCOMP'13 - The 2013 International Conference on Bioinformatics & Computational Biology

11.1.3. Misc.

Beat Wolf, Pierre Kuonen, Thomas Dandekar, *Multilevel parallelism in sequence alignment using a streaming approach*, Second International Workshop on Sustainable Ultrascale Computing Systems (NESUS 2015)

Beat Wolf, Monney Loïc, Pierre Kuonen *FriendComputing: Organic application centric distributed computing*, Second International Workshop on Sustainable Ultrascale Computing Systems (NESUS 2015)

Beat Wolf, Pierre Kuonen, *Distributed programming using POP-Java*, Doctoral Workshop on Distributed Systems, 2013

Beat Wolf, Pierre Kuonen, David Atlan, *General purpose distributed DNA aligner*, Doctoral Workshop on Distributed Systems, 2012

11.2. Posters

Beat Wolf, Pierre Kuonen, David Atlan, Jonathan Stoppani, Davide Mazzoleni, Thomas Dandekar, *Safe variant annotation sharing across laboratories*, Variant Detection 2017

Beat Wolf, Pierre Kuonen, David Atlan, Marco Lourenço, Jonathan Stoppani, Thomas Dandekar, *Trusted Friend Computing: data mining federated OMICS knowledge source*, European Human Genetics Conference 2017

Juliane Lippert, S. Appenzeller, S. Steinhauer, Simone Rost, **Beat Wolf**, Anrea Gehrig, C.R. Müller, M. Fassnacht, C.L. Ronchi, *Identification of a molecular signature for prognostic classification and individualized cancer therapy in adrenocortical carcinoma*, GfH Jahrestagung, 2017

Michaela Hofrichter, Barbara Vona, R. Maroofian, Linda Schnapp, Julia Doll, Tabea Röder, I. Nanda, Barry Chioza, **Beat Wolf**, Wafaa Shehata-Dieler, Erdmute Kunstmann, Jörg-Gunther Schröder, Tobias Müller, Ulrich Zechner, Oliver Bartsch, Marcus Dittrich, Andrew Crosby, Thomas Haaf, *Genotype-Phenotype correlation-The many facets of heterogeneous hearing loss in the context of molecular*, GfH Jahrestagung, 2017

Beat Wolf, Pierre Kuonen, Thomas Dandekar, *Meta-alignment: Combining multiple sequence aligners to improve alignment quality*, European Human Genetics Conference 2016

Beat Wolf, Pierre Kuonen, Thomas Dandekar, *GensearchNGS : Integrating OMICs analysing and visualization*, Visualizing Biological Data, VIZBI 2016

Böck Julia, Haertle Lara, Appenzeller S, **Beat Wolf**, Schneider T., Sutter C, Haaf T.,

Deep bisulfite sequencing for quantification of constitutive epimutations in tumor suppressor genes, GfH Jahrestagung, 2016

Ann-Kathrin Zaum, Wolfram Kreß, **Beat Wolf**, Simone Rost, *Novel dominant case of de-novo mutation in Ullrich congenital muscular dystrophy* , GfH Jahrestagung, 2016

Pluta Nathalie, Kress Wolfram, Clemens R. Müller, **Beat Wolf**, Rost Simone *Recessive truncating mutations in the TTN gene of two patients with muscular dystrophies* , GfH Jahrestagung, 2016

Beat Wolf, Pierre Kuonen, Thomas Dandekar, *Building blocks*, Visualizing Biological Data, VIZBI 2014

Beat Wolf, Pierre Kuonen, Thomas Dandekar, *GNATY: A tools library for faster variant calling and coverage analysis* , German Conference on Bioinformatics 2015

Beat Wolf, Pierre Kuonen, Thomas Dandekar, David Atlan, *Speeding up NGS analysis through local and remote computing resources*, European Human Genetics Conference 2015

Beat Wolf, Pierre Kuonen, Thomas Dandekar, David Atlan, *GensearchNGS: Interactive variant analysis*, 13th International Symposium on Mutation in the Genome: detection, genome sequencing & interpretation, 2015

Ann-Kathrin Zaum, Simone Rost, **Beat Wolf**, Clemens R. Müller, Thomas Musacchio, Erdmute Kunstmann, Stephan Klebe, *Distal hereditary motor neuropathy due to BSCL2 mutation in a two generation family*, GfH Jahrestagung, 2015

J Elisa Bach, **Beat Wolf**, Johannes Oldenburg, Clemens R Müller, Simone Rost, *Deep intronic variants in the factor VIII gene*, GfH Jahrestagung, 2015

Beat Wolf, Pierre Kuonen, Thomas Dandekar, *Towards integrative family analysis on OMICs data for individual patient diagnostics*, European Human Genetics Conference 2014

Beat Wolf, Pierre Kuonen, Thomas Dandekar, *GensearchNGS-Viewer: A complete NGS data visualization experience*, Visualizing Biological Data, VIZBI 2014

Beat Wolf, Pierre Kuonen, Thomas Dandekar, *Everybody's uniqueness*, Visualizing Biological Data, VIZBI 2014

J Elisa Bach, Simone Rost, Anna-Lena Semmler, Kristl G Claeys, **Beat Wolf**, Wolfram Kress, Clemens R Müller, *NGS panel for diagnostics of myofibrillar myopathies*, GfH Jahrestagung, 2014

David Atlan, Tim Beck, J.Brookes, Raymond Dalglish, Owen Lancaster, **Beat Wolf**, *Connecting diagnostic labs: Cafe Variome and DNA sequencing software*, Joint meeting of EURenOmics, NeurOmics and RDConnect in Heidelberg, 2014

Beat Wolf, Pierre Kuonen, David Atlan, Thomas Dandekar, Johan T. den Dunnen *Store, align and explore your genome outside the Cloud, at home, on your PC*, European Human Genetics Conference 2013

Beat Wolf, Pierre Kuonen, Thomas Dandekar, *Next generation sequencing from raw data to mutation report*, European Human Genetics Conference 2011

Bibliography

- [AAA⁺15] Adam Auton, Gonçalo R. Abecasis, David M. Altshuler, Richard M. Durbin, David R. Bentley, Aravinda Chakravarti, Andrew G. Clark, and Donnelly: A global reference for human genetic variation. *Nature*, 526(7571):68–74, 2015, ISSN 0028-0836.
- [AG15] Clemens R. Müller Andrea Gehrig: Hereditary breast/ovarian cancer: A systematic screening of dna repair genes in 300 consecutive patients. Poster presentation, 2015.
- [AG16] Clemens R. Müller Andrea Gehrig, Ines Schmitt: Hereditary breast/ovarian cancer: a systematic screening of 94 cancer associated genes in 500 consecutive index cases. Poster presentation, 2016.
- [AGM⁺90] S F Altschul, W Gish, W Miller, E W Myers, and D J Lipman: Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–10, oct 1990, ISSN 0022-2836. <http://www.ncbi.nlm.nih.gov/pubmed/2231712>.
- [AJL⁺14] Bruce Alberts, Alexander Johnson, Julian Lewis, David Morgan, Martin Raff, Keith Roberts, and Peter Walter: *Molecular Biology of the Cell*. Taylor & Francis Ltd., 6st edition, 2014, ISBN 0815344643, 9780815344643.
- [Amd67] Gene M. Amdahl: Validity of the single processor approach to achieving large scale computing capabilities. In Proceedings of the April 18-20, 1967, Spring Joint Computer Conference, AFIPS '67 (Spring), pages 483–485, New York, NY, USA, 1967. ACM. <http://doi.acm.org/10.1145/1465482.1465560>.
- [AMS⁺97] S F Altschul, T L Madden, a a Schäffer, J Zhang, Z Zhang, W Miller, and D J Lipman: Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic acids research*, 25(17):3389–402, sep 1997, ISSN 0305-1048.
- [AMZ⁺15] Zeeshan Ahmed, Michel Mayr, Saman Zeeshan, Thomas Dandekar, Martin J. Mueller, and Agnes Fekete: Lipid-Pro: a computational lipid identification solution for untargeted lipidomics on data-independent acquisition tandem mass spectrometry platforms., volume 31. Apr 2015. <http://www.ncbi.nlm.nih.gov/pubmed/25433698>.
- [And04] David P. Anderson: 3. BOINC: A system for public-resource computing and storage. *Proceedings - IEEE/ACM International Workshop on Grid Computing*, pages 4–10, 2004, ISSN 15505510.
- [APH15] Simon Anders, Paul Theodor Pyl, and Wolfgang Huber: HTSeq-A Python framework to work with high-throughput sequencing data. *Bioinformatics*, 31(2):166–169, 2015, ISSN 14602059.

- [APPA15] José M. Abuín, Juan C. Pichel, Tomás F. Pena, and Jorge Amigo: BigBWA: Approaching the Burrows-Wheeler aligner to Big Data technologies. *Bioinformatics*, 31(24):4003–4005, 2015, ISSN 14602059.
- [ARH12] Simon Anders, Alejandro Reyes, and Wolfgang Huber: Detecting differential usage of exons from RNA-seq data. *Genome Research*, 22(10):2008–2017, 2012, ISSN 10889051.
- [ASP⁺10] Ivan a Adzhubei, Steffen Schmidt, Leonid Peshkin, Vasily E Ramensky, Anna Gerasimova, Peer Bork, Alexey S Kondrashov, and Shamil R Sunyaev: A method and server for predicting damaging missense mutations. *Nature methods*, 7(4):248–9, apr 2010, ISSN 1548-7105.
- [BAW⁺16] Julia Böck, S. Appenzeller, Beat Wolf, T. Schneier, C. Sutter, and T. Haaf: Deep bisulfite sequencing for quantification of constitutive epimutations in tumor suppressor genes. Poster presentation, 2016.
- [BCD⁺15] J. A. Blake, K. R. Christie, M. E. Dolan, H. J. Drabkin, D. P. Hill, L. Ni, D. Sitnikov, S. Burgess, T. Buza, C. Gresham, F. McCarthy, L. Pillai, H. Wang, S. Carbon, H. Dietze, S. E. Lewis, C. J. Mungall, M. C. Munoz-Torres, M. Feuermann, P. Gaudet, S. Basu, R. L. Chisholm, R. J. Dodson, P. Fey, H. Mi, P. D. Thomas, A. Muruganujan, S. Poudel, J. C. Hu, S. A. Aleksander, B. K. McIntosh, D. P. Renfro, D. A. Siegele, H. Attrill, N. H. Brown, S. Tweedie, J. Lomax, D. Osumi-Sutherland, H. Parkinson, P. Roncaglia, R. C. Lovering, P. J. Talmud, S. E. Humphries, P. Denny, N. H. Campbell, R. E. Foulger, M. C. Chibucos, M. Gwinn Giglio, H. Y. Chang, R. Finn, M. Fraser, A. Mitchell, G. Nuka, S. Pesseat, A. Sangrador, M. Scheremetjew, S. Y. Young, R. Stephan, M. A. Harris, S. G. Oliver, K. Rutherford, V. Wood, J. Bahler, A. Lock, P. J. Kersey, M. D. McDowall, D. M. Staines, M. Dwinell, M. Shimoyama, S. Laulederkind, G. T. Hayman, S. J. Wang, V. Petri, P. D’Eustachio, L. Matthews, R. Balakrishnan, G. Binkley, J. M. Cherry, M. C. Costanzo, J. Demeter, S. S. Dwight, S. R. Engel, B. C. Hitz, D. O. Inglis, P. Lloyd, S. R. Miyasato, K. Paskov, G. Roe, M. Simison, R. S. Nash, M. S. Skrzypek, S. Weng, E. D. Wong, T. Z. Berardini, D. Li, E. Huala, J. Argasinska, C. Arighi, A. Auchincloss, K. Axelsen, G. Argoud-Puy, A. Bateman, B. Bely, M. C. Blatter, C. Bonilla, L. Bougueleret, E. Boutet, L. Breuza, A. Bridge, R. Britto, C. Casals, E. Cibrian-Uhalte, E. Coudert, I. Cusin, P. Duek-Roggli, A. Estreicher, L. Famiglietti, P. Gane, P. Garmiri, A. Gos, N. Gruaz-Gumowski, E. Hatton-Ellis, U. Hinz, C. Hulo, R. Huntley, F. Jungo, G. Keller, K. Laiho, P. Lemercier, D. Lieberherr, A. Macdougall, M. Magrane, M. Martin, P. Masson, P. Mutowo, C. O’Donovan, I. Pedruzzi, K. Pichler, D. Poggioli, S. Poux, C. Rivoire, B. Roechert, T. Sawford, M. Schneider, A. Shypitsyna, A. Stutz, S. Sundaram, M. Tognolli, C. Wu, I. Xenarios, J. Chan, R. Kishore, P. W. Sternberg, K. Van Auken, H. M. Muller, J. Done, Y. Li, D. Howe, and M. Westerfeld: Gene ontology consortium: Going forward. *Nucleic Acids Research*, 43(D1):D1049–D1056, 2015, ISSN 13624962.
- [Bea12] Wolf Beat: Distributed DNA alignment , a stream based approach. In Doctoral Workshop on Distributed Systems, 2012, pages 6–8, 2012.

- [BF07] Daniele Bonacorsi and Tiziana Ferrari: WLCG Service Challenges and Tiered architecture in the LHC era. *Incontri di Fisica delle Alte Energie*, pages 365–368, 2007. <http://www.springerlink.com/index/n06508035wx05p00.pdf>.
- [BH95] Yoav Benjamini and Yosef Hochberg: Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, 57(1):289–300, 1995.
- [BKB⁺15] M Broman, I Kleinschnitz, J E Bach, S Rost, G Islander, and C R Müller: Next-generation DNA sequencing of a Swedish malignant hyperthermia cohort. *Clinical Genetics*, 88:1–5, 2015, ISSN 00099163.
- [BKBW14] Xavier Barrelet, Pierre Kuonen, Frédéric Bapst, and Beat Wolf: CloudADN-II. Technical report, 2014.
- [BKW16] Christophe Blanquet, Pierre Kuonen, and Beat Wolf: POP – DNA I, Distributed Object Programming for high performance DNA analysis. Technical report, 2016.
- [BNK⁺14] Desislava Boyanova, Santosh Nilla, Gunnar W. Klau, Thomas Dandekar, Tobias Müller, and Marcus Dittrich: Functional module search in protein networks based on semantic similarity improves the analysis of proteomics data., volume 13. Jul 2014. <http://www.ncbi.nlm.nih.gov/pubmed/24807868>.
- [Boc12] Christoph Bock: Analysing and interpreting DNA methylation data. *Nature Reviews Genetics*, 13(10):705–719, 2012, ISSN 1471-0056. <http://dx.doi.org/10.1038/nrg3273>.
- [BP63] Thomas Bayes and Price Phil: An Essay towards Solving a Problem in the Doctrine of Chances. By the Late Rev. Mr. Bayes, F. R. S. Communicated by Mr. Price, in a Letter to John Canton, A. M. F. R. S. *Philosophical Transactions of the Royal Society of London*, 53:370–418, 1763.
- [BRM⁺05] Christoph Bock, Sabine Reither, Thomas Mikeska, Martina Paulsen, Jörn Walter, and Thomas Lengauer: BiQ Analyzer: Visualization and quality control for DNA methylation data from bisulfite sequencing. *Bioinformatics*, 21(21):4067–4068, 2005, ISSN 13674803.
- [BRS⁺14] J Elisa Bach, Simone Rost, Anna Lena Semmler, G Kristl Claeys, Beat Wolf, Wolfram Kress, and Clemens R Müller: Ngs panel for diagnostics of myofibrillar myopathies. Poster presentation, 2014.
- [BS12] Yuval Benjamini and Terence P. Speed: Summarizing and correcting the GC content bias in high-throughput sequencing. *Nucleic Acids Research*, 40(10):1–14, 2012, ISSN 03051048.
- [BSS95] J K Bonfield, K f Smith, and R Staden: A new DNA sequence assembly program. *Nucleic acids research*, 23(24):4992–4999, 1995, ISSN 0305-1048.
- [BTD⁺16] J. Borgmann, F. Tuttelmann, B. Dworniczak, A. Ropke, H. W. Song, S. Kliesch, M. F. Wilkinson, S. Laurentino, and J. Gromoll: The human RHOX

- gene cluster: target genes and functional analysis of gene variants in infertile men. *Hum. Mol. Genet.*, Sep 2016.
- [BW94] M Burrows and D.J. Wheeler: A block-sorting lossless data compression algorithm. 1994. <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.141.5254>.
- [BWO⁺15a] J Elisa Bach, Beat Wolf, Johannes Oldenburg, Clemens R Müller, and Simone Rost: Deep intronic variants in the factor viii gene. Poster presentation, 2015.
- [BWO⁺15b] J Elisa Bach, Beat Wolf, Johannes Oldenburg, Clemens R Müller, and Simone Rost: Identification of deep intronic variants in 15 haemophilia A patients by next generation sequencing of the whole factor VIII gene. *Thrombosis and Haemostasis*, pages 1–11, 2015.
- [BYD⁺16] Robert Buels, Eric Yao, Colin M Diesh, Richard D Hayes, Monica Munoz-Torres, Gregg Helt, David M Goodstein, Christine G Elsik, Suzanna E Lewis, Lincoln Stein, and Ian H Holmes: JBrowse: a dynamic web platform for genome visualization and analysis. *Genome Biology*, 17(1):1–12, 2016, ISSN 1474-760X. <http://dx.doi.org/10.1186/s13059-016-0924-1>.
- [CAB⁺15] Fiona Cunningham, M. Ridwan Amode, Daniel Barrell, Kathryn Beal, Konstantinos Billis, Simon Brent, Denise Carvalho-Silva, Peter Clapham, Guy Coates, Stephen Fitzgerald, Laurent Gil, Carlos García Girón, Leo Gordon, Thibaut Hourlier, Sarah E. Hunt, Sophie H. Janacek, Nathan Johnson, Thomas Juettemann, Andreas K. Kähäri, Stephen Keenan, Fergal J. Martin, Thomas Maurel, William McLaren, Daniel N. Murphy, Rishi Nag, Bert Overduin, Anne Parker, Mateus Patricio, Emily Perry, Miguel Pignatelli, Harpreet Singh Riat, Daniel Sheppard, Kieron Taylor, Anja Thormann, Alessandro Vullo, Steven P. Wilder, Amonida Zadissa, Bronwen L. Aken, Ewan Birney, Jennifer Harrow, Rhoda Kinsella, Matthieu Muffato, Magali Ruffier, Stephen M J Searle, Giulietta Spudich, Stephen J. Trevanion, Andy Yates, Daniel R. Zerbino, and Paul Flicek: Ensembl 2015. *Nucleic Acids Research*, 43(D1):D662–D669, 2015, ISSN 13624962.
- [CFG⁺09] Peter J A Cock, Christopher J. Fields, Naohisa Goto, Michael L. Heuer, and Peter M. Rice: The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. *Nucleic Acids Research*, 38(6):1767–1771, 2009, ISSN 03051048.
- [CGT14] Xin Bei V. Chan, Shi Min S. Goh, and Ngiap Chuan Tan: Subjects with colour vision deficiency in the community: what do primary care physicians need to know? *Asia Pacific Family Medicine*, 13(1):1–10, 2014, ISSN 1447-056X. <http://dx.doi.org/10.1186/s12930-014-0010-3>.
- [Cha88] K. Mani Chandy: Parallel Program Design: A Foundation. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1988, ISBN 0-201-05866-9.
- [Chr05] Chris Christmas, Rowan; Avila-Campillo, Iliana; Bolouri, Hamid; Schwikowski, Benno; Anderson, Mark; Kelley, Ryan; Landys, Nerius;

- Workman, Chris; Ideker, Trey; Cerami, Ethan; Sheridan, Rob; Bader, Gary D.; Sander: Cytoscape: a software environment for integrated models of biomolecular interaction networks. *American Association for Cancer Research Education Book*, (Karp 2001):12–16, 2005, ISSN 1088-9051.
- [CHS04] Denis Caromel, Ludovic Henrio, and Bernard Paul Serpette: Asynchronous and deterministic objects. *SIGPLAN Not.*, 39(1):123–134, January 2004, ISSN 0362-1340. <http://doi.acm.org/10.1145/982962.964012>.
- [CLZ⁺13] Wei Chen, Bingshan Li, Zhen Zeng, Serena Sanna, Carlo Sidore, Fabio Busonero, Hyun Min Kang, Yun Li, R Abecasis, North Carolina, Chapel Hill, and North Carolina: Genotype calling and haplotyping in parent-offspring trios. pages 1–11, 2013.
- [CMT⁺16] Ana Conesa, Pedro Madrigal, Sonia Tarazona, David Gomez-cabrero, Alejandra Cervera, Andrew Mcpherson, Wojciech Szcze, Daniel J Gaffney, Laura L Elo, and Xuegong Zhang: A survey of best practices for RNA-seq data analysis. pages 1–19, 2016.
- [CS10] Valentin Clément and Christian Senn: Bachelor Thesis 2010 Technical Report. 2010.
- [CSRM12] Aniruddha Chatterjee, Peter a. Stockwell, Euan J. Rodger, and Ian M. Morrison: Comparison of alignment software for genome-wide bisulphite sequence data. *Nucleic Acids Research*, 40(10):1–8, 2012, ISSN 03051048.
- [CT09] Francesco Cesarini and Simon Thompson: ERLANG Programming. O’Reilly Media, Inc., 1st edition, 2009, ISBN 0596518188, 9780596518189.
- [CWC12] Alexandra Chittka, Yannick Wurm, and Lars Chittka: Epigenetics: The making of ant castes. *Current Biology*, 22(19):R835–R838, 2012, ISSN 09609822. <http://dx.doi.org/10.1016/j.cub.2012.07.045>.
- [DA00] Johan T Den Dunnen and Stylianos E Antonarakis: Mutation Nomenclature Extensions and Suggestions to Describe Complex Mutations : A Discussion. 12:7–12, 2000.
- [DAA⁺11] Petr Danecek, Adam Auton, Goncalo Abecasis, Cornelis A. Albers, Eric Banks, Mark A. DePristo, Robert E. Handsaker, Gerton Lunter, Gabor T. Marth, Stephen T. Sherry, Gilean McVean, and Richard Durbin: The variant call format and VCFtools. *Bioinformatics*, 27(15):2156–2158, 2011, ISSN 13674803.
- [DBP⁺11] Mark A DePristo, Eric Banks, Ryan Poplin, Kiran V Garimella, Jared R Maguire, Christopher Hartl, Anthony A Philippakis, Guillermo del Angel, Manuel A Rivas, Matt Hanna, Aaron McKenna, Tim J Fennell, Andrew M Kernytsky, Andrey Y Sivachenko, Kristian Cibulskis, Stacey B Gabriel, David Altshuler, and Mark J Daly: A framework for variation discovery and genotyping using next-generation DNA sequencing data. *Nature genetics*, 43(5):491–498, 2011.

- [DDS⁺13] Alexander Dobin, Carrie a. Davis, Felix Schlesinger, Jorg Drenkow, Chris Zaleski, Sonali Jha, Philippe Batut, Mark Chaisson, and Thomas R. Gingeras: STAR: Ultrafast universal RNA-seq aligner. *Bioinformatics*, 29(1):15–21, 2013, ISSN 13674803.
- [DG08] Jeffrey Dean and Sanjay Ghemawat: MapReduce: simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, 2008, ISSN 0001-0782. <http://doi.acm.org/10.1145/1327452.1327492>.
- [DHL⁺09] François Olivier Desmet, Dalil Hamroun, Marine Lalande, Gwenaëlle Collod-Bérout, Mireille Claustres, and Christophe Bérout: Human Splicing Finder: An online bioinformatics tool to predict splicing signals. *Nucleic Acids Research*, 37(9):1–14, 2009, ISSN 03051048.
- [DM98] Leonardo Dagum and Ramesh Menon: Openmp: an industry standard api for shared-memory programming. *Computational Science & Engineering, IEEE*, 5(1):46–55, 1998.
- [DRA⁺13] Marie Agnès Dillies, Andrea Rau, Julie Aubert, Christelle Hennequet-Antier, Marine Jeanmougin, Nicolas Servant, Céline Keime, Nicolas Servant Marot, David Castel, Jordi Estelle, Gregory Guernec, Bernd Jagla, Luc Jouneau, Denis Laloë, Caroline Le Gall, Brigitte Schaëffer, Stéphane Le Crom, Mickaël Guedj, and Florence Jaffrézic: A comprehensive evaluation of normalization methods for Illumina high-throughput RNA sequencing data analysis. *Briefings in Bioinformatics*, 14(6):671–683, 2013, ISSN 14675463.
- [EbBS⁺96] Michael R Emmert-buck, Robert F Bonner, Paul D Smith, Rodrigo F Chuaqui, Zhengping Zhuang, Seth R Goldstein, Rhonda A Weiss, and Lance A Liotta: Laser Capture Microdissection. 274(November):998–1001, 1996.
- [EN14] Yaniv Erlich and Arvind Narayanan: Routes for breaching and protecting genetic privacy. *Nature Reviews Genetics*, 15(6):409–421, 2014, ISSN 1471-0056.
- [FBG⁺15] Simon A. Forbes, David Beare, Prasad Gunasekaran, Kenric Leung, Nidhi Bindal, Harry Boutselakis, Minjie Ding, Sally Bamford, Charlotte Cole, Sari Ward, Chai Yin Kok, Mingming Jia, Tisham De, Jon W. Teague, Michael R. Stratton, Ultan McDermott, and Peter J. Campbell: COSMIC: Exploring the world’s knowledge of somatic mutations in human cancer. *Nucleic Acids Research*, 43(D1):D805–D811, 2015, ISSN 13624962.
- [Fel13] Sarah Feldkirchner: Proteomische Einzelfaser-Analysen Myofibrillärer Myopathien mit Hilfe der Laser Dissektions- Mikroskopie. 2013.
- [For94] Message P Forum: Mpi: A message-passing interface standard. Technical report, Knoxville, TN, USA, 1994.
- [FUR⁺15] Adam Frankish, Barbara Uszczyńska, Graham R S Ritchie, Jose M Gonzalez, Dmitri Pervouchine, Robert Petryszak, Jonathan M Mudge, Nuno Fonseca, Alvis Brazma, Roderic Guigo, and Jennifer Harrow: Comparison of GENCODE and RefSeq gene annotation and the impact of reference geneset on

- variant effect prediction. *BMC genomics*, 16 Suppl 8(Suppl 8):S2, 2015, ISSN 1471-2164. <http://www.ncbi.nlm.nih.gov/pubmed/26110515>.
- [GBO⁺05] Jochen Graw, Hans Hermann Brackmann, Johannes Oldenburg, Reinhard Schneppenheim, Michael Spannagl, and Rainer Schwaab: Haemophilia A: from mutation analysis to new therapies. *Nature reviews. Genetics*, 6(6):488–501, 2005, ISSN 1471-0056.
- [Gen09] Craig Gentry: Fully homomorphic encryption using ideal lattices. *Proceedings of the 41st annual ACM symposium on Symposium on theory of computing STOC 09*, 19(September):169, 2009, ISSN 07378017. <http://portal.acm.org/citation.cfm?doid=1536414.1536440>.
- [GMG⁺13] Melissa Gymrek, Amy L. McGuire, David Golan, Eran Halperin, and Yaniv Erlich: Identifying personal genomes by surname inference. *Science*, 339(6117):321–324, 2013. <http://www.sciencemag.org/content/339/6117/321.abstract>.
- [GMM16] Sara Goodwin, John D. McPherson, and W. Richard McCombie: Coming of age: ten years of next-generation sequencing technologies. *Nature Reviews Genetics*, 17(6):333–351, 2016, ISSN 1471-0056. <http://www.nature.com/doifinder/10.1038/nrg.2016.49>.
- [Goo10] Leo Goodstadt: Ruffus: A lightweight python library for computational pipelines. *Bioinformatics*, 26(21):2778–2779, 2010, ISSN 13674803.
- [Got82] Osamu Gotoh: An improved algorithm for matching biological sequences. *Journal of Molecular Biology*, 162(3):705–708, 1982, ISSN 00222836.
- [GPHE⁺15] Jamal Ghoumid, Florence Petit, Muriel Holder-Espinasse, Anne Sophie Jourdain, José Guerra, Anne Dieux-Coeslier, Martin Figeac, Nicole Porchet, Sylvie Manouvrier-Hanu, and Fabienne Escande: Nail–Patella Syndrome: clinical and molecular data in 55 families raising the hypothesis of a genetic heterogeneity. *European Journal of Human Genetics*, (October 2014):1–7, 2015, ISSN 1018-4813. <http://www.nature.com/doifinder/10.1038/ejhg.2015.77>.
- [GRH05] Belinda Giardine, Cathy Riemer, and RC Hardison: Galaxy: a platform for interactive large-scale genome analysis. *Genome Research*, 15:1451–1455, 2005. <http://genome.cshlp.org/content/15/10/1451.short>.
- [GZM⁺15] Shengjie Gao, Dan Zou, Likai Mao, Quan Zhou, Wenlong Jia, Yi Huang, Shancen Zhao, Gang Chen, Song Wu, Dongdong Li, Fei Xia, Huafeng Chen, Maoshan Chen, Torben F Ørntoft, Lars Bolund, and Karina D Sørensen: SMAP: a streamlined methylation analysis pipeline for bisulfite sequencing. *GigaScience*, 4(1):29, 2015, ISSN 2047-217X. <http://www.gigasciencejournal.com/content/4/1/29>.
- [Hay98] R. H. Haynes: Heritable variation and mutagenesis at early international congresses of genetics, 1998. ISSN 00166731. <http://www.genetics.org/content/148/4/1419.full.pdf>.

- [Hay15] Ertica Check Hayden: Pint-sized DNA sequencer impresses first users Reality check for fossil-fuel divestment. *Nature*, 521(May):15–16, 2015, ISSN 0028-0836.
- [HBS73] Carl Hewitt, Peter Bishop, and Richard Steiger: A universal modular actor formalism for artificial intelligence. In Proceedings of the 3rd International Joint Conference on Artificial Intelligence, IJCAI'73, pages 235–245, San Francisco, CA, USA, 1973. Morgan Kaufmann Publishers Inc. <http://dl.acm.org/citation.cfm?id=1624775.1624804>.
- [HFG⁺12] Jennifer Harrow, Adam Frankish, Jose M. Gonzalez, Electra Tapanari, Mark Diekhans, Felix Kokocinski, Bronwen L. Aken, Daniel Barrell, Amonida Zadissa, Stephen Searle, If Barnes, Alexandra Bignell, Veronika Boychenko, Toby Hunt, Mike Kay, Gaurab Mukherjee, Jeena Rajan, Gloria Despacio-Reyes, Gary Saunders, Charles Steward, Rachel Harte, Michael Lin, Cédric Howald, Andrea Tanzer, Thomas Derrien, Jacqueline Chrast, Nathalie Walters, Suganthi Balasubramanian, Baikang Pei, Michael Tress, Jose Manuel Rodriguez, Iakes Ezkurdia, Jeltje Van Baren, Michael Brent, David Hausler, Manolis Kellis, Alfonso Valencia, Alexandre Reymond, Mark Gerstein, Roderic Guigó, and Tim J. Hubbard: GENCODE: The reference human genome annotation for the ENCODE project. *Genome Research*, 22(9):1760–1774, 2012, ISSN 10889051.
- [HNG⁺15] Michaela A.H. Hofrichter, Indrajit Nanda, Jens Gräf, Jörg Schröder, Wafaa Shehata-Dieler, Barbara Vona, and Thomas Haaf: A Novel de novo Mutation in CEACAM16 Associated with Postlingual Hearing Impairment. *Molecular Syndromology*, 6(4):156–163, 2015, ISSN 1661-8769. <http://www.karger.com/?doi=10.1159/000439576>.
- [HPL⁺12] Tamara Hansmann, Galyna Pliushch, Monika Leubner, Patricia Kroll, Daniela Endt, Andrea Gehrig, Sabine Preisler-Adams, Peter Wieacker, and Thomas Haaf: Constitutive promoter methylation of BRCA1 and RAD51C in patients with familial ovarian cancer and early-onset sporadic breast cancer. *Human Molecular Genetics*, 21(21):4669–4679, 2012, ISSN 09646906.
- [HPS⁺10] Yun Huang, William A. Pastor, Yinghua Shen, Mamta Tahiliani, David R. Liu, and Anjana Rao: The behaviour of 5-hydroxymethylcytosine in bisulfite sequencing. *PLoS ONE*, 5(1):1–9, January 2010. <http://dx.doi.org/10.1371/journal.pone.0008888>.
- [HTN14] Joseph Henson, German Tischler, and Zemin Ning: Europe PMC Funders Group Next-generation sequencing and large genome assemblies. 13(8):901–915, 2014.
- [HTS⁺08] Bastiaan T Heijmans, Elmar W Tobi, Aryeh D Stein, Hein Putter, Gerard J Blauw, Ezra S Susser, P Eline Slagboom, and L H Lumey: Persistent epigenetic differences associated with prenatal exposure to famine in humans. *Proceedings of the National Academy of Sciences of the United States of America*, 105(44):17046–9, 2008, ISSN 1091-6490.

- [Hul14] Sara Chandros Hull: The unintended implications of blurring the line between research and clinical care in a genomic age. 11:285–295, 2014.
- [HWK⁺15] Gareth Highnam, Jason J Wang, Dean Kusler, Justin Zook, Vinaya Vijayan, Nir Leibovich, and David Mittelman: ARTICLE An analytical framework for optimizing variant discovery from personal genomes. *Nature Communications*, 6:1–6, 2015. <http://dx.doi.org/10.1038/ncomms7275>.
- [Jar05] Thomas Jarvie: Next generation sequencing technologies. *Drug Discovery Today: Technologies*, 2(3):255 – 260, 2005, ISSN 1740-6749. <http://www.sciencedirect.com/science/article/pii/S1740674905000466>.
- [JFM⁺15] Miten Jain, Ian T Fiddes, Karen H Miga, Hugh E Olsen, Benedict Paten, and Mark Akeson: Improved data analysis for the MinION nanopore sequencer. 12(4), 2015.
- [JHZ⁺92] T Jordan, I Hanson, D Zaletayev, S Hodgson, J Prosser, A Seawright, N Hastie, and V van Heyningen: The human PAX6 gene is mutated in two patients with aniridia. *Nature genetics*, 1(5):328–32, 1992, ISSN 1061-4036. <http://www.ncbi.nlm.nih.gov/pubmed/1302030>.
- [KA11] Felix Krueger and Simon R. Andrews: Bismark: A flexible aligner and methylation caller for Bisulfite-Seq applications. *Bioinformatics*, 27(11):1571–1572, 2011, ISSN 13674803.
- [KBL16] Pierre Kuonen, Mathias Bavay, and Michael Lehning: POP-C++ and Alpine3D: Petition for a New HPC Approach. 2016.
- [KCB14] Pierre Kuonen, Valentin Clément, and Frédéric Bapst: Securing the Grid using Virtualization The ViSaG Model. (c):49–54, 2014.
- [KDM⁺14] Sebastian Köhler, Sandra C. Doelken, Christopher J. Mungall, Sebastian Bauer, Helen V. Firth, Isabelle Bailleul-Forestier, Graeme C M Black, Danielle L. Brown, Michael Brudno, Jennifer Campbell, David R. Fitzpatrick, Janan T. Eppig, Andrew P. Jackson, Kathleen Freson, Marta Girdea, Ingo Helbig, Jane A. Hurst, Johanna Jähn, Laird G. Jackson, Anne M. Kelly, David H. Ledbetter, Sahar Mansour, Christa L. Martin, Celia Moss, Andrew Mumford, Willem H. Ouwehand, Soo Mi Park, Erin Rooney Riggs, Richard H. Scott, Sanjay Sisodiya, Steven Van Vooren, Ronald J. Wapner, Andrew O M Wilkie, Caroline F. Wright, Anneke T. Vulto-Van Silfhout, Nicole De Leeuw, Bert B A De Vries, Nicole L. Washington, Cynthia L. Smith, Monte Westerfield, Paul Schofield, Barbara J. Ruef, Georgios V. Gkoutos, Melissa Haendel, Damian Smedley, Suzanna E. Lewis, and Peter N. Robinson: The Human Phenotype Ontology project: Linking molecular biology and disease through phenotype data. *Nucleic Acids Research*, 42(D1):966–974, 2014, ISSN 03051048.
- [KFM⁺12] Augustine Kong, Michael L. Frigge, Gisli Masson, Soren Besenbacher, Patrick Sulem, Gisli Magnusson, Sigurjon a. Gudjonsson, Asgeir Sigurdsson, Adalbjorg Aslaug Jonasdottir, Adalbjorg Aslaug Jonasdottir, Wendy S. W. Wong, Gunnar Sigurdsson, G. Bragi Walters, Stacy Steinberg, Hannes Helgason,

- Gudmar Thorleifsson, Daniel F. Gudbjartsson, Agnar Helgason, Olafur Th. Magnusson, Unnur Thorsteinsdottir, and Kari Stefansson: Rate of de novo mutations and the importance of father's age to disease risk. *Nature*, 488(7412):471–475, 2012, ISSN 0028-0836. <http://dx.doi.org/10.1038/nature11396>.
- [KGGK94] Vipin Kumar, Ananth Grama, Anshul Gupta, and George Karypis: Introduction to Parallel Computing: Design and Analysis of Algorithms. Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA, 1994, ISBN 0-8053-3170-0.
- [KHN09] Prateek Kumar, Steven Henikoff, and Pauline C Ng: Predicting the effects of coding non-synonymous variants on protein function using the SIFT algorithm. *Nature protocols*, 4(7):1073–81, jan 2009, ISSN 1750-2799. <http://www.ncbi.nlm.nih.gov/pubmed/19561590>.
- [KKFA12] Felix Krueger, Benjamin Kreck, Andre Franke, and Simon R Andrews: DNA methylome analysis using short bisulfite sequencing data. *Bioinformatics*, 9(2):145–51, 2012, ISSN 1548-7105. <http://www.ncbi.nlm.nih.gov/pubmed/22290186>.
- [KPT⁺13] Daehwan Kim, Geo Pertea, Cole Trapnell, Harold Pimentel, Ryan Kelley, and Steven L Salzberg: TopHat2: accurate alignment of transcriptomes in the presence of insertions, deletions and gene fusions. *Genome biology*, 14(4):R36, apr 2013, ISSN 1465-6914. <http://www.ncbi.nlm.nih.gov/pubmed/23618408>.
- [KR12] Johannes Köster and Sven Rahmann: Snakemake—a scalable bioinformatics workflow engine. *Bioinformatics*, 28(19):2520–2522, 2012, ISSN 13674803.
- [KSA⁺10] Kazuhiko Komatsu, Katsuto Sato, Yusuke Arai, Kentaro Koyama, Hiroyuki Takizawa, and Hiroaki Kobayashi: Evaluating Performance and Portability of OpenCL Programs. *Science And Technology*, 2:52, 2010. <http://vecpar.fe.up.pt/2010/workshops-iWAPT/Komatsu-Sato-Arai-Koyama-Takizawa-Kobayashi.pdf>.
- [KSF⁺02] W James Kent, Charles W Sugnet, Terrence S Furey, Krishna M Roskin, Tom H Pringle, Alan M Zahler, and David Haussler: The Human Genome Browser at UCSC The Human Genome Browser at UCSC. *Genome Research*, 12:996–1006, 2002.
- [KSK⁺16] Minoru Kanehisa, Yoko Sato, Masayuki Kawashima, Miho Furumichi, and Mao Tanabe: KEGG as a reference resource for gene and protein annotation. *Nucleic Acids Research*, 44(D1):D457–D462, 2016, ISSN 0305-1048. <http://nar.oxfordjournals.org/lookup/doi/10.1093/nar/gkv1070>.
- [KWS⁺17] Meik Kunz, Beat Wolf, Harald Schulze, David Atlan, Thorsten Walles, Heike Walles, and Thomas Dandekar: Non-coding rnas in lung cancer: Contribution of bioinformatics analysis to the development of non-invasive diagnostic tools. *Genes*, 8(1):8, 2017, ISSN 2073-4425. <http://www.mdpi.com/2073-4425/8/1/8>.

- [KWS⁺ss] Meik Kunz, Beat Wolf, Harald Schulze, David Atlan, Thorsten Walles, Heike Walles, and Thomas Dandekar: Non-coding RNAs in lung cancer: Potential as non-invasive diagnostic tools and bioinformatics analysis approaches. *Genes*, in press.
- [KZL⁺12] Daniel C. Koboldt, Qunyuan Zhang, David E. Larson, Dong Shen, Michael D. McLellan, Ling Lin, Christopher A. Miller, Elaine R. Mardis, Li Ding, and Richard K. Wilson: VarScan 2: Somatic mutation and copy number alteration discovery in cancer by exome sequencing. *Genome Research*, 22(3):568–576, 2012.
- [Lar15] Mirjam Larsen: Zur genetischen Heterogenität der Muskeldystrophien : alternative genetische Ursachen der Myotonen Dystrophie und FSHD. PhD thesis, 2015. <https://opus.bibliothek.uni-wuerzburg.de/frontdoor/index/index/docId/12343>.
- [LBA⁺15] Owen Lancaster, Tim Beck, David Atlan, Morris Swertz, Dhiwagaran Thangavelu, Colin Veal, Raymond Dagleish, and Anthony J. Brookes: Cafe Variome: General-Purpose Software for Making Genotype-Phenotype Data Discoverable in Restricted or Open Access Contexts. *Human Mutation*, 36(10):957–964, 2015, ISSN 10981004.
- [LD09] Heng Li and Richard Durbin: Fast and accurate short read alignment with Burrows–Wheeler transform. *Bioinformatics (Oxford, England)*, 25(14):1754–60, jul 2009, ISSN 1367-4811.
- [Lek15] Monkol Lek: Analysis of protein-coding genetic variation in 60,706 humans. pages 1–26, 2015, ISSN 1098-6596.
- [LFH⁺01] Cecilia S. L. Lai, Simon E. Fisher, Jane A. Hurst, Faraneh Vargha-Khadem, and Anthony P. Monaco: A forkhead-domain gene is mutated in a severe speech and language disorder. *Nature*, 413(6855):519–523, 2001, ISSN 0028-0836.
- [LG11] Gerton Lunter and Martin Goodson: Stampy: A statistical algorithm for sensitive and fast mapping of Illumina sequence reads. *Genome Research*, 21(6):936–939, 2011, ISSN 10889051.
- [LGC⁺16] Yanzhu Lin, Kseniya Golovkina, Zhen Xia Chen, Hang Noh Lee, Yazmin L Serrano Negron, Hina Sultana, Brian Oliver, and Susan T Harbison: Comparison of normalization and differential expression analyses using RNA-Seq data from 726 individual *Drosophila melanogaster*. *BMC genomics*, 17(1):28, 2016, ISSN 1471-2164. <http://bmcbgenomics.biomedcentral.com/articles/10.1186/s12864-015-2353-z>.
- [LH10] Heng Li and Nils Homer: A survey of sequence alignment algorithms for next-generation sequencing. *Briefings in bioinformatics*, 11(5):473–83, sep 2010, ISSN 1477-4054.

- [LHA14] Michael I Love, Wolfgang Huber, and Simon Anders: Moderated estimation of fold change and dispersion for RNA-Seq data with DESeq2 Moderated estimation of fold change and dispersion for RNA-Seq data with DESeq2. 2014.
- [LHO⁺15] T. Laver, J. Harrison, P.A. O'Neill, K. Moore, A. Farbos, K. Paszkiewicz, and D.J. Studholme: Assessing the performance of the Oxford Nanopore Technologies MinION. *Biomolecular Detection and Quantification*, 3:1–8, 2015, ISSN 22147535. <http://linkinghub.elsevier.com/retrieve/pii/S2214753515000224>.
- [LHW⁺09] Heng Li, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, and Richard Durbin: The Sequence Alignment/Map format and SAMtools. *Bioinformatics (Oxford, England)*, 25(16):2078–2079, aug 2009, ISSN 1367-4811.
- [Li13] Heng Li: Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. *arXiv preprint arXiv*, 00(00):3, 2013, ISSN 2169-8287. <http://arxiv.org/abs/1303.3997>.
- [LKS⁺16] Mirjam Larsen, Wolfram Kress, Benedikt Schoser, Ute Hehr, Clemens R Müller, and Simone Rost: Identification of variants in MBNL1 in patients with a myotonic dystrophy-like phenotype. *Eur J Hum Genet*, may 2016. <http://dx.doi.org/10.1038/ejhg.2016.41>.
- [LLB⁺01] E S Lander, L M Linton, B Birren, C Nusbaum, M C Zody, J Baldwin, K Devon, K Dewar, and M Doyle: Initial sequencing and analysis of the human genome. *Nature*, 409(6822):860–921, 2001, ISSN 0028-0836. <http://www.ncbi.nlm.nih.gov/pubmed/11237011>.
- [LLB⁺16] Melissa J Landrum, Jennifer M Lee, Mark Benson, Garth Brown, Chen Chao, Shanmuga Chitipiralla, Baoshan Gu, Jennifer Hart, Douglas Hoffman, Jeffrey Hoover, Wonhee Jang, Kenneth Katz, Michael Ovetsky, George Riley, Amanjeev Sethi, Ray Tully, Ricardo Villamarin-Salomon, Wendy Rubinstein, and Donna R Maglott: ClinVar: public archive of interpretations of clinically relevant variants. *Nucleic acids research*, 44(D1):D862–8, 2016, ISSN 1362-4962 (ELECTRONIC).
- [LLKW08] Ruiqiang Li, Yingrui Li, Karsten Kristiansen, and Jun Wang: SOAP: short oligonucleotide alignment program. *Bioinformatics (Oxford, England)*, 24(5):713–4, mar 2008, ISSN 1367-4811. <http://www.ncbi.nlm.nih.gov/pubmed/18227114>.
- [LP83] Michalowsky Lesley A and Jones Peter A: Dna Methylation and Differentiation. *Environmental Health Perspectives*, 80:189–197, 1983.
- [LRE⁺15] Mirjam Larsen, Simone Rost, Nady El Hajj, Andreas Ferbert, Marcus Deschauer, Maggie C Walter, Benedikt Schoser, Pawel Tacik, Wolfram Kress, and Clemens R Müller: Diagnostic approach for FSHD revisited: SMCHD1 mutations cause FSHD2 and act as modifiers of disease severity in FSHD1.

- European Journal of Human Genetics*, 23(6):808–816, 2015, ISSN 1018-4813. <http://www.nature.com/doi/10.1038/ejhg.2014.191>.
- [LS12a] Ben Langmead and Steven L Salzberg: Fast gapped-read alignment with Bowtie 2. *Nature methods*, 9(4):357–9, apr 2012, ISSN 1548-7105. <http://www.ncbi.nlm.nih.gov/pubmed/22388286>.
- [LS12b] Y. Liu and B. Schmidt: Long read alignment based on maximal exact match seeds. *Bioinformatics*, 28(18):i318–i324, sep 2012, ISSN 1367-4803. <http://bioinformatics.oxfordjournals.org/cgi/doi/10.1093/bioinformatics/bts414>.
- [LS14] Yongchao Liu and Bertil Schmidt: CUSHAW2-GPU: Empowering faster gapped short-read alignment using GPU computing. *IEEE Design and Test*, 31(1):31–39, 2014, ISSN 21682356.
- [LSN⁺07] Samuel Levy, Granger Sutton, Pauline C. Ng, Lars Feuk, Aaron L. Halpern, Brian P. Walenz, Nelson Axelrod, Jiaqi Huang, Ewen F. Kirkness, Gennady Denisov, Yuan Lin, Jeffrey R. MacDonald, Andy Wing Chun Pang, Mary Shago, Timothy B. Stockwell, Alexia Tsiamouri, Vineet Bafna, Vikas Bansal, Saul A. Kravitz, Dana A. Busam, Karen Y. Beeson, Tina C. McIntosh, Karin A. Remington, Josep F. Abril, John Gill, Jon Borman, Yu Hui Rogers, Marvin E. Frazier, Stephen W. Scherer, Robert L. Strausberg, and J. Craig Venter: The diploid genome sequence of an individual human. *PLoS Biology*, 5(10):2113–2144, 2007, ISSN 15449173.
- [LSSP09] Stefan M. Larson, Christopher D. Snow, Michael Shirts, and Vijay S. Pande: Folding@Home and Genome@Home: Using distributed computing to tackle previously intractable problems in computational biology. page 31, 2009. <http://arxiv.org/abs/0901.0866>.
- [LST⁺08] T. W. Lam, W. K. Sung, S. L. Tam, C. K. Wong, and S. M. Yiu: Compressed indexing and local alignment of DNA. *Bioinformatics*, 24(6):791–797, 2008, ISSN 1367-4803. <http://bioinformatics.oxfordjournals.org/cgi/doi/10.1093/bioinformatics/btn032>.
- [LTPS09] Ben Langmead, Cole Trapnell, Mihai Pop, and Steven L Salzberg: Ultra-fast and memory-efficient alignment of short DNA sequences to the human genome. *Genome biology*, 10(3):R25, jan 2009, ISSN 1465-6914.
- [Mar06] Elaine R. Mardis: Anticipating the 1,000 dollar genome. *Genome biology*, 7:112, 2006, ISSN 1465-6914.
- [Mar08] Elaine R. Mardis: Next-Generation DNA Sequencing Methods. *Annual Review of Genomics and Human Genetics*, 9(1):387–402, 2008, ISSN 1527-8204. <http://www.annualreviews.org/doi/abs/10.1146/annurev.genom.9.081307.164359>.
- [MHB⁺10] Aaron McKenna, Matthew Hanna, Eric Banks, Andrey Sivachenko, Kristian Cibulskis, Andrew Kernytsky, Kiran Garimella, David Altshuler, Stacey Gabriel, Mark Daly, and Mark A. DePristo: The genome analysis toolkit:

- A mapreduce framework for analyzing next-generation dna sequencing data. *Genome Research*, 20(9):1297–1303, 2010.
- [MKA14] Deepti Mittal, Damandeep Kaur, and Ashish Aggarwal: Secure Data Mining in Cloud Using Homomorphic Encryption. *2014 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, pages 1–7, 2014. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7015496>.
- [MKPW16] Davide Mazzoleni, Pierre Kuonen, Benoit Perroud, and Beat Wolf: POP-DNA II, Parallel Object Programming in a Hadoop environment. Technical report, 2016.
- [MKW⁺13] Andrea Marcacci, Pierre Kuonen, Beat Wolf, David Atlan, and Pierre alain Mettraux: Cloud computing pour l ’ analyse de séquences d ’ ADN humain. Technical report, 2013.
- [Mon14] Loic Monney: Sharing computing power through a network of friends. Technical Report August, 2014.
- [MPR⁺10] William McLaren, Bethan Pritchard, Daniel Rios, Yuan Chen, Paul Flicek, and Fiona Cunningham: Deriving the consequences of genomic variants with the Ensembl API and SNP Effect Predictor. *Bioinformatics*, 26(16):2069–2070, 2010, ISSN 13674803.
- [MSB⁺13] Iain Milne, Gordon Stephen, Micha Bayer, Peter J A Cock, Leighton Pritchard, Linda Cardle, Paul D. Shawand, and David Marshall: Using tablet for visual exploration of second-generation sequencing data. *Briefings in Bioinformatics*, 14(2):193–202, 2013, ISSN 14675463.
- [MWM⁺08] A. Mortazavi, B. A. Williams, K. McCue, L. Schaeffer, and B. Wold: Mapping and quantifying mammalian transcriptomes by RNA-Seq. *Nat. Methods*, 5(7):621–628, Jul 2008.
- [MZÜ⁺16] Thomas Musacchio, Ann Kathrin Zaum, Nurcan Üçeyler, Claudia Sommer, Nora Pfeifroth, Karlheinz Reiners, Erdmute Kunstmann, Jens Volkmann, Simone Rost, and Stephan Klebe: Als and mmn mimics in patients with bscl2 mutations: the expanding clinical spectrum of spg17 hereditary spastic paraplegia. *Journal of Neurology*, pages 1–10, 2016, ISSN 1432-1459. <http://dx.doi.org/10.1007/s00415-016-8301-2>.
- [NBGS08] John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron: Scalable parallel programming with cuda. *Queue*, 6(2):40–53, March 2008, ISSN 1542-7730. <http://doi.acm.org/10.1145/1365490.1365500>.
- [NGU04] Tuan Anh NGUYEN: An object-oriented model for adaptive high-performance computing on the computational grid. 3079, 2004.
- [NHG⁺17] Adam M Novak, Glenn Hickey, Erik Garrison, Sean Blum, Abram Connelly, Alexander Dilthey, Jordan Eizenga, M. A. Saleh Elmohamed, Sally Guthrie, André Kahles, Stephen Keenan, Jerome Kelleher, Deniz Kural,

- Heng Li, Michael F Lin, Karen Miga, Nancy Ouyang, Goran Rakocevic, Maciek Smuga-Otto, Alexander Wait Zaranek, Richard Durbin, Gil McVean, David Haussler, and Benedict Paten: Genome graphs. *bioRxiv*, 2017. <http://biorxiv.org/content/early/2017/01/18/101378>.
- [NK02] T A Nguyen and Pierre Kuonen: A Model of Dynamic Parallel Objects for Metacomputing. In The 2002 International Conference on Parallel and Distributed Processing Techniques and Applications, BIOCOMP'13, 2002.
- [NK07] Tuan Anh Nguyen and Pierre Kuonen: Programming the grid with pop-c++. *Future Gener. Comput. Syst.*, 23(1):23–30, January 2007, ISSN 0167-739X. <http://dx.doi.org/10.1016/j.future.2006.04.012>.
- [NKC⁺16] Julie Zhouli Ni, Natallia Kalinava, Esteban Chen, Alex Huang, Thi Trinh, and Sam Guoping Gu: A transgenerational role of the germline nuclear rna1 pathway in repressing heat stress-induced transcriptional activation in *c. elegans*. *Epigenetics & Chromatin*, 9(1):1–15, 2016. <http://dx.doi.org/10.1186/s13072-016-0052-x>.
- [NP16] Wolfram Kress Clemens R. Müller Simone Rost Natalie Pluta, Gitta Emmert: Analysis of 37 / 65 muscle genes in 300 patients with neuromuscular diseases. Poster presentation, 2016.
- [NPR15] Marius Nicolae, Sudipta Pathak, and Sanguthevar Rajasekaran: LFQC: A lossless compression algorithm for FASTQ files. *Bioinformatics*, 31(20):3276–3281, 2015, ISSN 14602059.
- [NW70] Saul B Needleman and Christian D Wunsch: A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, 1970, ISSN 0022-2836. <http://www.sciencedirect.com/science/article/pii/0022283670900574>.
- [Oa04] Martin Odersky and al.: An Overview of the Scala Programming Language. Technical Report IC/2004/64, EPFL, Lausanne, Switzerland, 2004.
- [OB13] C. S. Oehmen and D. J. Baxter: ScalaBLAST 2.0: Rapid and robust BLAST calculations on multiprocessor systems. *Bioinformatics*, pages 1–2, jan 2013, ISSN 1367-4803. <http://bioinformatics.oxfordjournals.org/cgi/doi/10.1093/bioinformatics/btt013>.
- [OWB⁺16] Nuala A O’Leary, Mathew W Wright, J Rodney Brister, Stacy Ciufu, Diana Haddad, Rich McVeigh, Bhanu Rajput, Barbara Robbertse, Brian Smith-White, Danso Ako-Adjei, Alexander Astashyn, Azat Badretdin, Yiming Bao, Olga Blinkova, Vyacheslav Brover, Vyacheslav Chetvernin, Jinna Choi, Eric Cox, Olga Ermolaeva, Catherine M Farrell, Tamara Goldfarb, Tripti Gupta, Daniel Haft, Eneida Hatcher, Wratko Hlavina, Vinita S Joardar, Vamsi K Kodali, Wenjun Li, Donna Maglott, Patrick Masterson, Kelly M McGarvey, Michael R Murphy, Kathleen O’Neill, Shashikant Pujar, Sanjida H Rangwala, Daniel Rausch, Lillian D Riddick, Conrad Schoch, Andrei Shkeda, Susan S Storz, Hanzhen Sun, Francoise Thibaud-Nissen, Igor Tolstoy, Raymond E

- Tully, Anjana R Vatsan, Craig Wallin, David Webb, Wendy Wu, Melissa J Landrum, Avi Kimchi, Tatiana Tatusova, Michael DiCuccio, Paul Kitts, Terence D Murphy, and Kim D Pruitt: Reference sequence (RefSeq) database at NCBI: current status, taxonomic expansion, and functional annotation. *Nucleic Acids Research*, 44(D1):D733–D745, 2016, ISSN 0305-1048. <http://nar.oxfordjournals.org/lookup/doi/10.1093/nar/gkv1189>.
- [PMK14] Rob Patro, Stephen M Mount, and Carl Kingsford: Sailfish enables alignment-free isoform quantification from RNA-seq reads using lightweight algorithms. *Nature biotechnology*, 32(5):462–464, 2014, ISSN 1546-1696. <http://www.ncbi.nlm.nih.gov/pubmed/24752080>.
- [PP13] Armando J Pinho and Diogo Pratas: MFCompress: a compression tool for FASTA and multi-FASTA data. *Bioinformatics (Oxford, England)*, pages 1–2, nov 2013, ISSN 1367-4811. <http://www.ncbi.nlm.nih.gov/pubmed/24132931>.
- [PPK⁺07] Ruth Pordes, Don Petravick, Bill Kramer, Doug Olson, Miron Livny, Alain Roy, Paul Avery, Kent Blackburn, Torre Wenaus, Frank Würthwein, Ian Foster, Rob Gardner, Mike Wilde, Alan Blatecky, John McGee, and Rob Quick: The open science grid. *Journal of Physics: Conference Series*, 78:012057, 2007, ISSN 1742-6588.
- [PTW01] P a Pevzner, H Tang, and M S Waterman: An Eulerian path approach to DNA fragment assembly. *Proceedings of the National Academy of Sciences of the United States of America*, 98(17):9748–9753, 2001, ISSN 00278424.
- [PWZM97] William R Pearson, Todd Wood, Zheng Zhang, and Webb Miller: Comparison of DNA sequences with protein sequences. *Genomics*, 46(1):24–36., 1997, ISSN 0888-7543.
- [QH10] Aaron R. Quinlan and Ira M. Hall: BEDTools: A flexible suite of utilities for comparing genomic features. *Bioinformatics*, 26(6):841–842, 2010.
- [R D08] R Development Core Team: R: A language and environment for statistical computing. Technical report, Vienna, Austria, 2008. <http://www.R-project.org>, ISBN 3-900051-07-0.
- [RA15] Anthony Rhoads and Kin Fai Au: PacBio Sequencing and Its Applications. *Genomics, Proteomics and Bioinformatics*, 13(5):278–289, 2015, ISSN 22103244. <http://dx.doi.org/10.1016/j.gpb.2015.08.002>.
- [RAB⁺15] Sue Richards, Nazneen Aziz, Sherri Bale, David Bick, Soma Das, Julie Gastier-Foster, Wayne W. Grody, Madhuri Hegde, Elaine Lyon, Elaine Spector, Karl Voelkerding, and Heidi L. Rehm: Standards and guidelines for the interpretation of sequence variants: a joint consensus recommendation of the American College of Medical Genetics and Genomics and the Association for Molecular Pathology. *Genetics in Medicine*, 17(5):405–423, 2015, ISSN 1098-3600. <http://dx.doi.org/10.1038/gim.2015.30>.

- [RBN⁺14] Simone Rost, Elisa Bach, Cordula Neuner, Indrajit Nanda, Sandra Dysek, Reginald E Bittner, Alexander Keller, Oliver Bartsch, Robert Mlynski, Thomas Haaf, Clemens R Müller, and Erdmute Kunstmann: Novel form of X-linked nonsyndromic hearing loss with cochlear malformation caused by a mutation in the type IV collagen gene COL4A6. *European Journal of Human Genetics*, 22(2):208–215, 2014, ISSN 1018-4813. <http://www.nature.com/doifinder/10.1038/ejhg.2013.108>.
- [RKO⁺14] Peter N Robinson, Sebastian Köhler, Anika Oellrich, Sanger Mouse Genetics Project, Kai Wang, Christopher J Mungall, Suzanna E Lewis, Nicole Washington, Sebastian Bauer, Dominik Seelow, Peter Krawitz, Christian Gilissen, Melissa Haendel, and Damian Smedley: Improved exome prioritization of disease genes through cross-species phenotype comparison. *Genome research*, 24(2):340–348, 2014, ISSN 1088-9051. <http://europepmc.org/articles/PMC3912424>.
- [RPW⁺12] Andreas Ruppen, Jacques Pasquier, Jean Frédéric Wagen, Beat Wolf, and Raphael Guye: A WoT approach to eHealth: case study of a hospital laboratory alert escalation system. *Proceedings of the Third International Workshop on the Web of Things*, page 6, 2012.
- [RTC⁺14] Thomas Rolland, Murat Taşan, Benoit Charlotiaux, Samuel J. Pevzner, Quan Zhong, Nidhi Sahni, Song Yi, Irma Lemmens, Celia Fontanillo, Roberto Mosca, Atanas Kamburov, Susan D. Ghiassian, Xinping Yang, Lila Ghamdari, Dawit Balcha, Bridget E. Begg, Pascal Braun, Marc Brehme, Martin P. Broly, Anne Ruxandra Carvunis, Dan Convery-Zupan, Roser Corominas, Jasmin Coulombe-Huntington, Elizabeth Dann, Matija Dreze, Amélie Dricot, Changyu Fan, Eric Franzosa, Fana Gebreab, Bryan J. Gutierrez, Madeleine F. Hardy, Mike Jin, Shuli Kang, Ruth Kiros, Guan Ning Lin, Katja Luck, Andrew Macwilliams, Jörg Menche, Ryan R. Murray, Alexandre Palagi, Matthew M. Poulin, Xavier Rambout, John Rasla, Patrick Reichert, Viviana Romero, Elie Ruysinck, Julie M. Sahalie, Annemarie Scholz, Akash A. Shah, Amitabh Sharma, Yun Shen, Kerstin Spirohn, Stanley Tam, Alexander O. Tejada, Shelly A. Trigg, Jean Claude Twizere, Kerwin Vega, Jennifer Walsh, Michael E. Cusick, Yu Xia, Albert László Barabási, Lilia M. Iakoucheva, Patrick Aloy, Javier De Las Rivas, Jan Tavernier, Michael A. Calderwood, David E. Hill, Tong Hao, Frederick P. Roth, and Marc Vidal: A proteome-scale map of the human interactome network. *Cell*, 159(5):1212–1226, 2014, ISSN 10974172.
- [RZ14] Xavier Rogé and Xuegong Zhang: RNAseqViewer: Visualization tool for RNA-Seq data. *Bioinformatics*, 30(6):891–892, 2014, ISSN 13674803.
- [SB14] Markus D Siegelin and Alain C Borczuk: Epidermal growth factor receptor mutations in lung adenocarcinoma. *Laboratory investigation; a journal of technical methods and pathology*, 94(2):129–37, 2014, ISSN 1530-0307. <http://www.ncbi.nlm.nih.gov/pubmed/24378644>.
- [SBR⁺14] Joachim Schessl, Elisa Bach, Simone Rost, Sarah Feldkirchner, Christiana Kubny, Stefan Müller, Franz Georg Hanisch, Wolfram Kress, and Benedikt

- Schoser: Novel recessive myotilin mutation causes severe myofibrillar myopathy. *Neurogenetics*, 15(3):151–6, 2014, ISSN 1364-6753. <http://www.ncbi.nlm.nih.gov/pubmed/24928145>.
- [Sch09a] Michael C Schatz: CloudBurst: highly sensitive read mapping with MapReduce. *Bioinformatics (Oxford, England)*, 25(11):1363–9, jun 2009, ISSN 1367-4811.
- [SCH⁺09b] Andrew D. Smith, Wen Yu Chung, Emily Hodges, Jude Kendall, Greg Hannon, James Hicks, Zhenyu Xuan, and Michael Q. Zhang: Updates to the RMAP short-read mapping software. *Bioinformatics*, 25(21):2841–2842, 2009, ISSN 13674803.
- [SD13] C Soneson and M Delorenzi: A comparison of methods for differential expression analysis of RNA-seq data. *BMC Bioinformatics*, 14:91, 2013, ISSN BMC BIOINFORMATICS. http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?cmd=Retrieve&db=PubMed&dopt=Citation&list_uids=23497356.
- [SGS10] John E. Stone, David Gohara, and Guochun Shi: Opencl: A parallel programming standard for heterogeneous computing systems. *IEEE Des. Test*, 12(3):66–73, May 2010, ISSN 0740-7475. <http://dx.doi.org/10.1109/MCSE.2010.69>.
- [Sis16] Mario Sisto: VisuDNA. Technical report, 2016.
- [SMW⁺02] J. P. Schouten, C. J. McElgunn, R. Waaijer, D. Zwijnenburg, F. Diepvens, and G. Pals: Relative quantification of 40 nucleic acid sequences by multiplex ligation-dependent probe amplification. *Nucleic Acids Res.*, 30(12):e57, Jun 2002.
- [SNY13] Shuying Sun, Aaron Noviski, and Xiaoqing Yu: MethyQA: a pipeline for bisulfite-treated methylation sequencing quality assessment. *BMC bioinformatics*, 14(1):259, 2013, ISSN 1471-2105. <http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3765750&tool=pmcentrez&rendertype=abstract>.
- [SR96] S. Schiaffino and C. Reggiani: Molecular diversity of myofibrillar proteins: gene regulation and functional significance. *Physiological Reviews*, 76(2):371–423, 1996, ISSN 0031-9333. <http://physrev.physiology.org/content/76/2/371>.
- [Sri10] Sriram Srinivasan: Kilim: A server framework with lightweight actors, isolation types and zero-copy messaging. Technical Report UCAM-CL-TR-769, University of Cambridge, Computer Laboratory, February 2010. <http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-769.pdf>.
- [SSB⁺14] Anna Lena Semmler, Sabrina Sacconi, J Bach, Claus Liebe, Jan Bürmann, Rudolf a Kley, Andreas Ferbert, Roland Anderheiden, Peter Van den Bergh, Jean Jacques Martin, Peter De Jonghe, Eva Neuen-Jacob, Oliver Müller, Marcus Deschauer, Markus Bergmann, J Schröder, Matthias Vorgerd, Jörg B

- Schulz, Joachim Weis, Wolfram Kress, and Kristl G Claeys: Unusual multisystemic involvement and a novel BAG3 mutation revealed by NGS screening in a large cohort of myofibrillar myopathies. *Orphanet journal of rare diseases*, 9(1):121, 2014, ISSN 1750-1172. <http://www.ncbi.nlm.nih.gov/pubmed/25208129>.
- [SW81] T.F. Smith; and M S Waterman: Identification of Common Molecular Subsequences. *J. Mol. Biol.*, 147:195–197, 1981, ISSN 00222836.
- [SWK⁺01] S T Sherry, M H Ward, M Kholodov, J Baker, L Phan, E M Smigielski, and K Sirotkin: dbSNP: the NCBI database of genetic variation. *Nucleic acids research*, 29(1):308–11, 2001, ISSN 1362-4962.
- [TBS⁺06] Mike Tyers, Ashton Breitkreutz, Chris Stark, Teresa Reguly, Lorrie Boucher, and Bobby Joe Breitkreutz: BioGRID: a general repository for interaction datasets. *Nucl. Acids Res.*, 34(suppl_1):D535–539, 2006, ISSN 1362-4962.
- [Tin15] Comp U Ting: Cloud cover protects gene data. *Nature*, 519:400–401, 2015, ISSN 1476-4687.
- [TRGP12] C Trapnell, A Roberts, L Goff, and G Pertea: Differential gene and transcript expression analysis of RNA-seq experiments with TopHat and Cufflinks. *Nature protocols*, 7(3):562–578, 2012. <http://www.nature.com/nprot/journal/v7/n3/abs/nprot.2012.016.html>.
- [TRM13] Helga Thorvaldsdóttir, James T Robinson, and Jill P Mesirov: Integrative Genomics Viewer (IGV): high-performance genomics data visualization and exploration. *Briefings in bioinformatics*, 14(2):178–92, mar 2013, ISSN 1477-4054.
- [TS06] Andrew S. Tanenbaum and Maarten van Steen: Distributed Systems: Principles and Paradigms (2Nd Edition). Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006, ISBN 0132392275.
- [Var13] Carlos A. Varela: Programming Distributed Computing Systems: A Foundational Approach. The MIT Press, 2013, ISBN 0262018985, 9780262018982.
- [VB12] Joris a Veltman and Han G Brunner: De novo mutations in human genetic disease. *Nature reviews. Genetics*, 13(8):565–75, 2012, ISSN 1471-0064. <http://www.ncbi.nlm.nih.gov/pubmed/22805709>.
- [VJC⁺13] Sven Verdoolaege, Juan Carlos Juega, Albert Cohen, José Ignacio Gómez, Christian Tenllado, and Francky Catthoor: Polyhedral parallel code generation for cuda. *ACM Trans. Archit. Code Optim.*, 9(4):54:1–54:23, January 2013, ISSN 1544-3566.
- [WANW14] Charles D Warden, Aaron W Adamson, Susan L Neuhausen, and Xiwei Wu: Detailed comparison of two popular variant calling packages for exome and targeted exon studies. *PeerJ*, 2:e600, 2014, ISSN 2167-8359. <http://dx.doi.org/10.7717/peerj.600>.

- [Whi12] Tom White: Hadoop: The Definitive Guide. O'Reilly Media, Inc., 2012, ISBN 1449311520, 9781449311520.
- [WK13a] Beat Wolf and Pierre Kuonen: A novel approach for heuristic pairwise DNA sequence alignment. In Proceedings of the 2013 International Conference on Bioinformatics & Computational Biology, BIOCOMP'13, 2013.
- [WK13b] Beat Wolf and Pierre Kuonen: Distributed programming using POP-Java. In Doctoral Workshop on Distributed Systems, pages 1–3, 2013.
- [WKD14a] Beat Wolf, Pierre Kuonen, and Thomas Dandekar: Everybodys uniqueness. Poster presentation, 2014.
- [WKD14b] Beat Wolf, Pierre Kuonen, and Thomas Dandekar: Gensearchngs-viewer: A complete ngs data visualization experience. Poster presentation, 2014.
- [WKD14c] Beat Wolf, Pierre Kuonen, and Thomas Dandekar: POP-Java: Parallélisme et distribution orienté objet. In Etienne Riviere Arnaud Tisserand Pascal Felber Laurent Philippe (editor): COMPAS 2014 : conférence en parallélisme, architecture et systèmes, Neuchâtel, Suisse, apr 2014. <http://hal.inria.fr/hal-00988568>.
- [WKD14d] Beat Wolf, Pierre Kuonen, and Thomas Dandekar: Towards integrative family analysis on omics data for individual patient diagnostics. Poster presentation, 2014.
- [WKD⁺14e] Beat Wolf, Pierre Kuonen, Thomas Dandekar, Haute École, and Spécialisée De Suisse: Comment reproduire les résultats de l'article : POP-Java : Parallélisme et distribution orienté objet. In Realis 2014 : Reproductibilité expérimentale pour l'informatique en parallélisme, architecture et système – 2ème édition, 2014.
- [WKD15] Beat Wolf, Pierre Kuonen, and Thomas Dandekar: Gnaty: A tools library for faster variant calling and coverage analysis. Poster presentation, 2015.
- [WKD16a] Beat Wolf, Pierre Kuonen, and Thomas Dandekar: Bioinformatics and Biomedical Engineering: 4th International Conference, IWBBIO 2016, Granada, Spain, April 20-22, 2016, Proceedings, chapter GNATY: Optimized NGS Variant Calling and Coverage Analysis, pages 446–454. Springer International Publishing, Cham, 2016, ISBN 978-3-319-31744-1. http://dx.doi.org/10.1007/978-3-319-31744-1_40.
- [WKD16b] Beat Wolf, Pierre Kuonen, and Thomas Dandekar: Building blocks. Poster presentation, 2016.
- [WKD16c] Beat Wolf, Pierre Kuonen, and Thomas Dandekar: Gensearchngs : Integrating omics analysis and visualization. Poster presentation, 2016.
- [WKDA15a] Beat Wolf, Pierre Kuonen, Thomas Dandekar, and David Atlan: DNaseq Workflow in a Diagnostic Context and an Example of a User Friendly Implementation. *BioMed research international*, 2015:403497, 2015, ISSN 2314-6141.

- [WKDA15b] Beat Wolf, Pierre Kuonen, Thomas Dandekar, and David Atlan: Gensearchings: Interactive variant analysis. Poster presentation, 2015.
- [WLH10] Kai Wang, Mingyao Li, and Hakon Hakonarson: ANNOVAR: functional annotation of genetic variants from high-throughput sequencing data. *Nucleic acids research*, 38(16):e164, 2010, ISSN 13624962.
- [WMK15] Beat Wolf, Loïc Money, and Pierre Kuonen: FriendComputing : Organic application centric distributed computing. In Nesus 2015, volume I, pages 1–3, 2015.
- [Wol11] Beat Wolf: Analysis and visualization of DNA sequences using cloud computing. Master’s thesis, University of Applied Sciences Western Switzerland, 2011.
- [XL09] Yuanxin Xi and Wei Li: BSMAP: whole genome bisulfite sequence MAPping program. *BMC bioinformatics*, 10:232, 2009, ISSN 1471-2105.
- [YAA⁺16] Andrew Yates, Wasiu Akanni, M. Ridwan Amode, Daniel Barrell, Konstantinos Billis, Denise Carvalho-Silva, Carla Cummins, Peter Clapham, Stephen Fitzgerald, Laurent Gil, Carlos García Girón, Leo Gordon, Thibaut Hourlier, Sarah E. Hunt, Sophie H. Janacek, Nathan Johnson, Thomas Juettemann, Stephen Keenan, Ilias Lavidas, Fergal J. Martin, Thomas Maurel, William McLaren, Daniel N. Murphy, Rishi Nag, Michael Nuhn, Anne Parker, Mateus Patricio, Miguel Pignatelli, Matthew Rahtz, Harpreet Singh Riat, Daniel Sheppard, Kieron Taylor, Anja Thormann, Alessandro Vullo, Steven P. Wilder, Amonida Zadissa, Ewan Birney, Jennifer Harrow, Matthieu Muffato, Emily Perry, Magali Ruffier, Giulietta Spudich, Stephen J. Trevanion, Fiona Cunningham, Bronwen L. Aken, Daniel R. Zerbino, and Paul Flicek: Ensembl 2016. *Nucleic acids research*, 44(D1):D710–D716, 2016, ISSN 1362-4962. <http://nar.oxfordjournals.org/lookup/doi/10.1093/nar/gkv1157>.
- [YMR⁺13] Yaping Yang, Donna M Muzny, Jeffrey G Reid, Matthew N Bainbridge, Alecia Willis, Patricia a Ward, Alicia Braxton, Joke Beuten, Fan Xia, Zhiyv Niu, Matthew Hardison, Richard Person, Mir Reza Bekheirnia, Magalie S Leduc, Amelia Kirby, Peter Pham, Jennifer Scull, Min Wang, Yan Ding, Sharon E Plon, James R Lupski, Arthur L Beaudet, Richard a Gibbs, and Christine M Eng: Clinical whole-exome sequencing for the diagnosis of mendelian disorders. *The New England journal of medicine*, 369(16):1502–11, 2013, ISSN 1533-4406. <http://www.ncbi.nlm.nih.gov/pubmed/24088041>.
- [YWSO10] Matthew D Young, Matthew J Wakefield, Gordon K Smyth, and Alicia Oshlack: Gene ontology analysis for RNA-seq: accounting for selection bias. *Genome biology*, 11(2):R14, 2010, ISSN 1465-6906.
- [ZFW⁺03] Barry R Zeeberg, Weimin Feng, Geoffrey Wang, May D Wang, Anthony T Fojo, Margot Sunshine, Sudarshan Narasimhan, David W Kane, William C Reinhold, Samir Lababidi, Kimberly J Bussey, Joseph Riss, J Carl Barrett, and John N Weinstein: GoMiner: a resource for biological interpretation of genomic and proteomic data. *Genome biology*, 4(4):R28, 2003, ISSN 1474-760X.

- [ZLL⁺11] Yan Zhang, Hongbo Liu, Jie Lv, Xue Xiao, Jiang Zhu, Xiaojuan Liu, Jianzhong Su, Xia Li, Qiong Wu, Fang Wang, and Ying Cui: QDMR: A quantitative method for identification of differentially methylated regions by entropy. *Nucleic Acids Research*, 39(9), 2011, ISSN 03051048.
- [ZLR14] Xiaobei Zhou, Helen Lindsay, and Mark D. Robinson: Robustly detecting differential expression in RNA sequencing data using observation weights. *Nucleic Acids Research*, 42(11), 2014, ISSN 13624962.
- [ZLY⁺15] Yongpeng Zhang, Linsen Li, Yanli Yang, Xiao Yang, Shan He, and Zexuan Zhu: Light-weight reference-based compression of FASTQ data. *BMC bioinformatics*, 16(1):188, 2015, ISSN 1471-2105.
- [ZS10] Arie Zackay and Christine Steinhoff: MethVisual - visualization and exploratory statistical analysis of DNA methylation profiles from bisulfite sequencing. *BMC research notes*, 3(1):337, 2010, ISSN 1756-0500.
- [ZWW⁺13] Min Zhao, Qingguo Wang, Quan Wang, Peilin Jia, and Zhongming Zhao: art%3A10.1186%2F1471-2105-14-S11-S1. 14(Suppl 11), 2013, ISSN 1471-2105.

Appendices

A. Artistic data visualization

This chapter presents artistic approaches to visualize genetic data which were created over the course of this thesis. Science is constantly seeking for answers to push the boundaries of human knowledge. To do this, science requires people interested in the subject and also public support for the science being done. An excellent to do this is art, as it can attract the interest of people normally not interested in a particular subject.

Over the course of this thesis the opportunity presented itself multiple times at the VIZBI conference (Visualizing Biological Data) to create an artistic visualization of genetic data. With one of the central points of the thesis being to lower the complexity of OMICs analysis this opportunity was fit perfectly into the theme.

And indeed, the two posters created for the VIZBI conference, *Everybody's uniqueness* [WKD14a] as well as *Building blocks* [WKD16b], were the works created during this thesis that generated most feedback and discussion. As the VIZBI conference format did not allow for a more indepth explanation of the submitted art, the rest of this chapter takes a closer look at those two posters.

A.0.1. Variant visualization

The human DNA is made out of 3 billion basepairs, organized in 23 chromosomes (not counting the mitochondrial DNA). Any two individuals differ on average in one out of 1000 bases, which amount to 3 million differences between any two people. Those are big, abstract numbers and not intuitive to understand. This is why the possibility was explored to represent this information in a more accessible way, without the pretension to be a useful tool in diagnostics.

The core concept of the idea was to create an artistic picture which is unique for every individual. It should also allow any individual, even without deep understanding of genetics, to learn something. This should be true in particular when comparing the picture of multiple individuals.

The idea to represent the genome of an individual as the sum of its variants came quite early in the process. Using the variants of an individual has the advantage of being a stable source of information and in theory no dependent on how the information was acquired. If an individual is sequenced multiple times during its lifetime using different technologies, it should, considering no sequencing artifacts, always result in the same set of variants.

Initially several ideas were explored, but representing the individual variant as a colored dot on the picture seemed to be the most promising approach. A variant has several properties, some describing the actual change in the genome, and some the functional consequences of said change. Base on the idea that a variant can be represented as a point on a picture, the question was which properties of the variant determine which property of the point.

A point can have several properties and for this project the following have been chosen:

- The x/y coordinates on the picture

- The radius which determines the size
- The color, which includes the alpha channel used for transparency

After initial trials, the position of the variant on the genome was chosen as the source of the x/y coordinates. The initial problem to solve was the layout of the picture. The first test used the genomic position of every variant as the seed for a random number generator. Every variant received a random, but reproducible position on the picture (meaning, two individuals with the same variant have a point at the same location). The change of the variant, for example A>G, determined the color of the point. This initial test did indeed produce unique pictures for every individual. But it was neither aesthetically pleasing to look at nor was it possible to get a better understanding of genetics out of it.

The second idea was to base the picture on a popular movie, “The Matrix”. With DNA being the source code of all life, it seemed rather fitting to use the emblematic picture used during the movie to represent the code of the matrix. But there were a couple of problems with that approach. The idea was to separate the variants, based on which chromosome they come from, into separate groups drawn as vertical strips on the picture. Probably the biggest issue is that not all chromosomes have the same size and thus also a different amount of variants. This led to a unbalanced picture which did not satisfy the criteria of being visually pleasing to look at.

While this particular layout was not successful, the idea of separating the variants based on their chromosomes was kept. The following idea, which turned out to be the one that worked best, was to position the variants on a more complex geometric shape than a line, the circle. The idea was to position all variants on circles, one circle for every chromosome, and the position on the circle would be determined by the position of the variant on the chromosome. This layout did indeed stay in the final version, of which a modified version can be found in figure A.1. The standard circle equation was used to position the variants where $x = \cos(p) * r$ and $y = \sin(p) * r$. p is defined as $p = (V_p * 2\pi) / C_l$ where V_p is the position of the variant on the chromosome and C_l the length of the chromosome. r is defined as $r = l * V_c$ where V_c is the chromosome number of the variant and l is defined as $l = R/23$ with R being the radius of the biggest circle.

The other question to solve was on how to color the individual points on the picture. Basing the color, as it was done initially, on the genomic change was one way to do it. But as the changes in the genome are more or less random, the colors used throughout the picture did not show any meaningful structure. By using not the genomic change, but the predicted consequence of the variant (one of the major annotations used during variant analysis), the structure of the genome and its changes became more apparent. In figure A.1 variants on splice sites are marked in red, stop codon additions or removals are orange and frameshifts are black. The other types of variants have various colors attributed to them, with blue being used for intragenetic variants. The size of the individual points is linked to the severeness of the prediction, with splice site variants being the biggest and intragenetic variants the smallest.

This coloring scheme immediately allows to appreciate the structure of the genome and its changes. The picture created for figure A.1 as well as the original poster is based on the variants found in the individual NA12882, from the Illumina Platinum trio. The data was aligned against the human reference version HG19. The variants are drawn in a counter clock wise based on the position of chromosome.

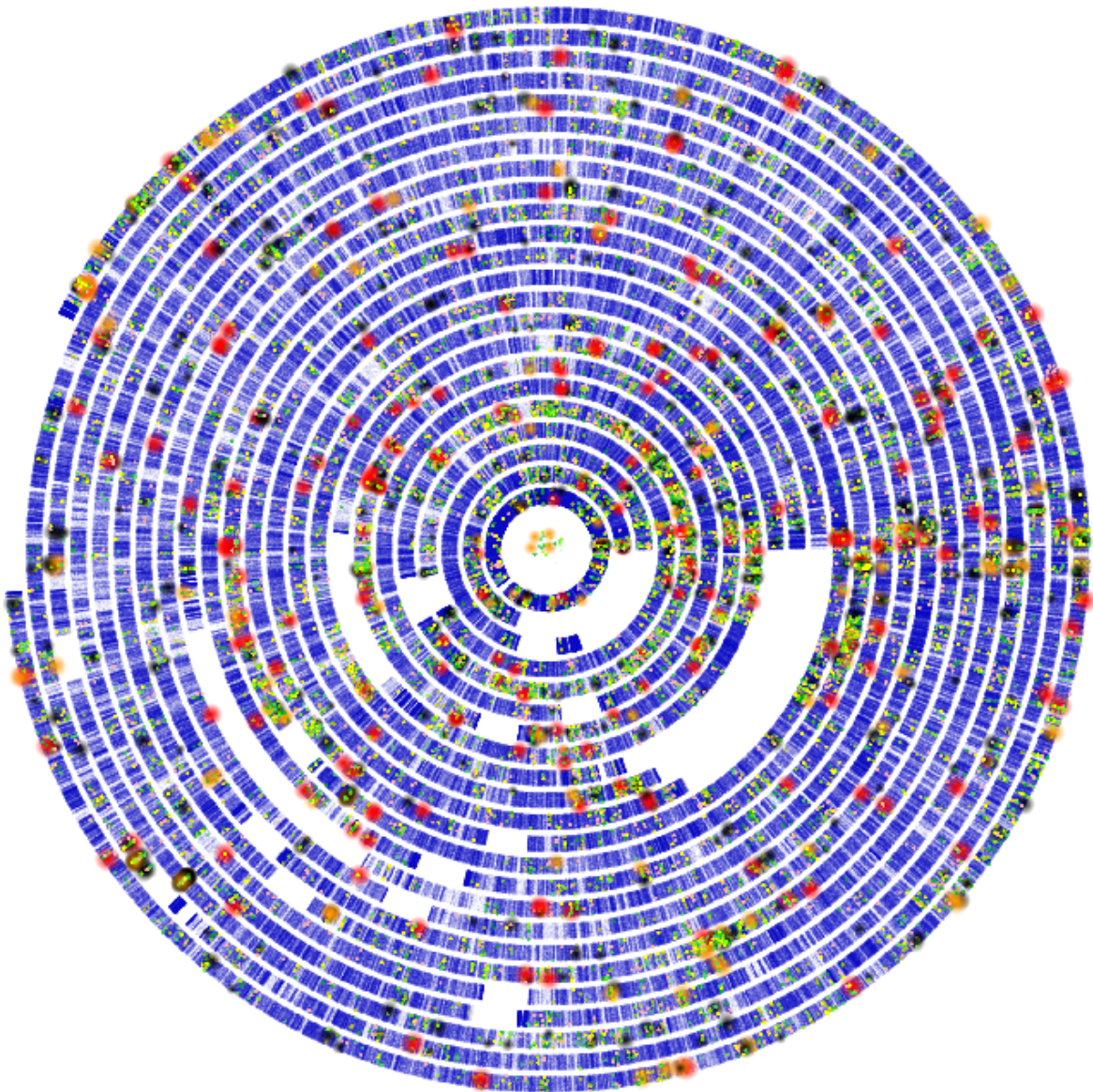


Fig. A.1.: Variant of the original poster [WKD14a], with a white background.

One of the first things that can be noticed are the large amount of blue points. This shows two things: The amount of variants which are outside of genes and also the amount of variants that fall into the least understood portions of the genome. The white stretches in the circles can also be easily seen. They show the parts of the human reference genome for which, at the moment of the creation of the HG19 reference genome, no information was known. NA12882 is a healthy individual, yet a surprising amount of potentially bad variants can be seen (red, orange and black). Those maybe benign observations for a geneticist can help people from outside the field to understand more about what DNA is and what makes everybody unique.

When combining the pictures created from multiple related people, other interesting observations can be made. While for space reasons only one example of such a picture is shown in this thesis, one can easily imagine the observations that can be made when looking at a trio of individuals (mother, father and child). Just by looking at the 3 images

it is possible to determine who the males and females are (because of the missing variants on the Y chromosome for females). By closer inspection it can then also be deduced who the parents are and who the child. One example is the mitochondrial DNA which is always inherited from the mother. If the child is male, it is easy to differentiate between the father and the son, just by comparing the variants on the mitochondrial DNA (at the center of the picture) with the mother. The picture that has the same variants as the mother is the child.

Those examples are just a few of the things that can be discovered when looking closely at those pictures. While those pictures are not intended to be perfectly accurate or a tool for serious scientific studies, they nonetheless render science and genetics more approachable for the general public. Being able to create a personalized work of art that is unique for every person was a highly rewarding and interesting experience. To facilitate the usage of this tool, it has been integrated into GensearchNGS to easily create an image of this type from any analysis that is done with the software.

A.0.2. Building blocks

The human genome consists of about 25'000 genes. All those genes collaborate in a complicated manner and specialize in a particular task. But what exactly a gene is and how it influences our body is hard to understand and it remains a very abstract notion for many. The poster *Building blocks* [WKD16b] is an attempt to make genes a little bit more approachable by giving them a “face”.

The underlying idea of the poster is to show how different genes are responsible for different organs in the body. The idea is also to show how the human body, and live in general, is made out of four basic nucleotides. This is where the idea was born to use the genetic sequence of different genes to draw organs with which they are associated. The four basic nucleotides were decided to be displayed using their standard colors, which is to say green for A (adenine), black for C (cytosine), red for T (thymine) and blue for G (guanine).

As there are four nucleotides, four organs were chosen to be displayed. While ultimately the choice of what organs to display was arbitrary, we believe that they represent four organs to which people can related and also recognize. The organs were first drawn as black and white pictures, which were then colored using a custom computer program using the coding sequence of four different genes.

The genes, were chosen for being important for everyone of those organs.

As genes, including their intronic DNA, can be very long, only the start of the coding sequence of the different genes were used. The genes which were chosen are the following:

PAX6 : PAX6 is a critical gene for eye development. Variations on this gene have been linked to aniridia [JHZ⁺92], which is a disorder which causes the absence of the iris for an individual.

FOXP2 : FOXP2 is a gene associated with the development of speak and language. Severe disorders in development of speech and language have been associated with this gene [LFH⁺01].

EGFR : EGFR, which stands for epidermal growth factor receptor, is a gene associated with lung cancer. 15-20% of certain types of lung cancers have been associated with modifications to this gene [SB14].

MYH6 : MYH6 is a gene which is important for cardiac muscles. It has been identified as one of the most transcribed genes inside the cardiac muscles [SR96].

The final result which has been presented at VIZBI 2016 can be seen in Figure A.2.

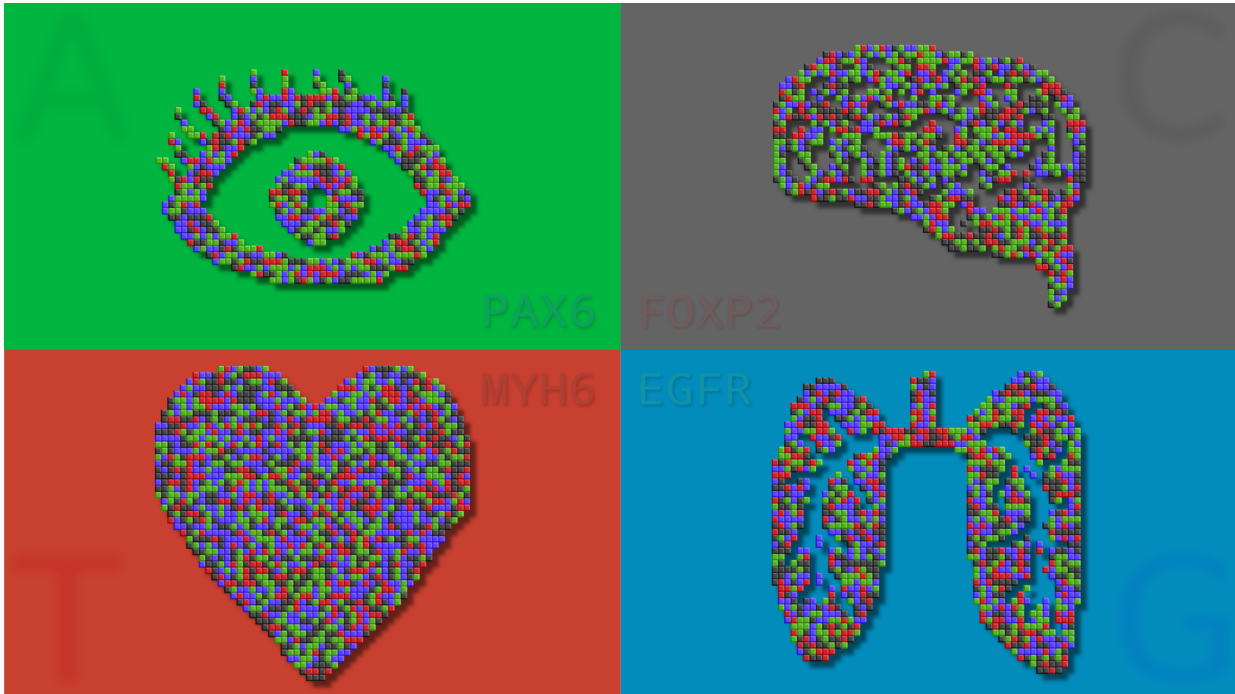


Fig. A.2.: Artistic poster submitted at VIZBI 2016 [WKD16b]

B. Meta-alignment

This appendix chapter contains the raw data from Section 7.2.

B.1. Simulated dataset

Tab. B.1.: Raw alignment values used for Figure 7.13

Recall	1	2	3	4	5	6	7	8	9	10	11
Bowtie2	100.00%	100.00%	99.93%	98.96%	95.99%	90.59%	83.30%	75.07%	66.54%	58.85%	52.19%
BWA	100.00%	100.00%	100.00%	99.86%	98.87%	95.96%	88.80%	75.41%	58.77%	43.26%	31.63%
Cushaw2	100.00%	99.94%	98.43%	92.87%	83.19%	70.63%	57.68%	45.46%	35.04%	27.17%	21.74%
Meta	100.00%	100.00%	100.00%	99.91%	99.28%	97.37%	93.03%	85.21%	75.03%	64.69%	55.74%
Meta 2	98.85%	98.17%	96.60%	93.16%	86.82%	77.55%	65.51%	51.63%	37.96%	26.88%	18.81%
Meta 3	96.84%	93.73%	88.21%	79.07%	67.06%	53.21%	39.33%	26.97%	17.22%	10.59%	6.37%

Tab. B.2.: Raw precision values used for Figure 7.14

Precision	1	2	3	4	5	6	7	8	9	10	11
Bowtie2	98.49%	96.07%	92.78%	89.11%	84.84%	79.83%	74.32%	68.93%	63.51%	58.33%	53.06%
BWA	98.49%	98.14%	97.40%	95.81%	93.42%	90.33%	85.55%	78.65%	70.17%	59.97%	48.24%
Cushaw2	97.20%	95.63%	92.96%	89.27%	84.99%	80.52%	76.14%	72.75%	70.09%	69.44%	70.63%
Meta	98.47%	98.19%	97.46%	95.89%	93.26%	89.72%	84.99%	79.24%	72.90%	66.71%	61.58%
Meta 2	99.38%	99.18%	98.89%	98.36%	97.53%	96.55%	95.46%	94.03%	92.38%	90.64%	88.34%
Meta 3	99.84%	99.76%	99.68%	99.52%	99.20%	98.75%	98.14%	97.07%	95.64%	93.93%	91.13%

C. Custom file formats

This chapter describes the various custom file formats used during this thesis.

C.1. Variants

This section describes the custom variant file format used in GensearchNGS to store variants. The default filename extension used is `.var`, which is an uncompressed text file.

Header

The header contains the general information of the file and consists of two lines. In C.1 an example of the header section is shown.

Listing C.1: Variant file header section

```
1 ##version=6
2 Chrom    Position          Ref      Var      Coverage+
   Coverage-    Reads+  Reads-  Quality ID      Pato      SIFT
   Poly2      MAF
```

The first line contains the version number of the file format, which allows for backward compatibility when reading older files. The second line contains the name of all columns present that will contain data for the stored variants.

Body

After the header section, the individual variants are stored, with one variant per line. Every variant has a list of attributes, which are separated by tabs. Table C.1 has a list of all those attributes.

Tab. C.1.: List of all attributes saved for a variant

Chrom	The name of the chromosome, ex: 12	String
Position	The position of the variant on the chromosome	Integer
Ref	The reference sequence at that position	String
Var	The sequence of the variant	String
Coverage+	Coverage at the position of the variant on the forward strand	Integer
Coverage-	Coverage at the position of the variant on the reverse strand	Integer
Reads+	Amount of reads supporting the variant on the forward strand	Integer
Reads-	Amount of reads supporting the variant on the reverse strand	Integer
Quality	Quality information based as reported by the variant caller	Double
ID	ID of the variant if present in a database, ex: rs1234	String
Pato	Patogenicity information, based on clinvar	String
SIFT	SIFT score of the variant	Double
Poly2	Polyphen 2 score of the variant	Double
MAF	Minor allele frequency of the variant	Double

C.2. Epigenetics

The information generated during methylation analysis of bisulfite sequencing data is stored in a custom fileformat. The file is a text file with the following structure, a header followed by the body.

Header

The header only contains the information about the file version information, on a line starting with the character #.

Body

The body of the file contains one entry per line. Every entry represents a genomic region, which can be an entire chromosome as well as a promoter region of a gene. The individual values are separated by tabs with a newline indicating the start of a new entry. Table C.2 shows the content of every line.

Tab. C.2.: List of all attributes saved for a variant

Name	The name of the genomic region	String
Gene	The name of gene associated with the region (can be empty)	String
Start	The start position of the genomic region	Integer
End	The end position of the genomic region	Integer
CpG	Amount of CpGs identified in the region	Long
Meth	Amount methylated CpGs identified in the region	Long
Normal	Amount non methylated CpGs identified in the region	Long
Other	Amount methylated non CpG Cs identified in the region	Long
Average	Average methylation of the sequences in that region	Double
0-25	Number of reads with a methylation of 0-25%	Long
25-50	Number of reads with a methylation of 25-50%	Long
50-75	Number of reads with a methylation of 50-75%	Long
75-100	Number of reads with a methylation of 75-100%	Long

C.3. Expression

While GensearchNGS supports the HTseq fileformat for expression analysis, we created a custom fileformat to store additional information not present in the HTseq fileformat.

By default, the fileformat is compressed using gzip. The fileformat is a binary fileformat, which speeds up the parsing.

Header

The header is structured as follows (Table C.3):

Tab. C.3.: Description of the fields found in the header

Version	The version of the files data format	Integer
Species	The name of the species	String
Reference	Version of the reference that was used	Integer
Chromosome	The name of the chromosome	String

Body

The body contains the individual regions that were analysed. Those can be on the gene, transcript or exon level.

Tab. C.4.: List of fields for every expressed region

Type	The type of the feature (G, T or E)	Char
Name	The name of the feature	String
ID	The ID of the feature	String
Start	First position of the feature (0 based)	Integer
Stop	Last position of the feature (0 based)	Integer
SubRegionLength	Length of all subregions (like exons)	Integer
Sequences	Number of sequences that map to this feature	Integer
Mismatches	Sequences that partially map to other features	Integer
Exlusive	Sequences where the start and endpoint map to this feature	Integer

D. Polling

D.1. User polling results

This Section contains the raw answers for the poll discussed in 9.3.

What is your profession?

Tab. D.1.: What is your profession?

Profession	Count
Geneticist	9
Biologist	3
Genetic counselor	1
Technical Assistant	1
Bio-informatician	1
PhD student	1

Tab. D.2.: How would you rate your general computer skills?

Skill	1	2	3	4	5	6
	0	1	2	10	2	1

Tab. D.3.: How long have you been using GensearchNGS?

Time	1 month	6 months	1 year	2+years
	2	1	3	10

Tab. D.4.: How often do use GensearchNGS?

Frequency	Once a month	Once a week	Daily	I don't use it regularly
	2	5	7	2

Tab. D.5.: How many samples do you analyse with GensearchNGS per year?

<u>Samples</u>
200
300
300
200
200
> 200
150
50
12
1000
80
36
350
200
17
400

Tab. D.6.: In what context to you use GensearchNGS?

<u>Context</u>	<u>Count</u>
Research	13
Diagnostics	12
Commercial	0
Education	1
Other	0

Tab. D.7.: What datasources do you analyse with GensearchNGS?

<u>Source</u>	<u>Count</u>
DNaseq	14
RNAseq	0
Bisulfite sequencing	2
Other	0

Tab. D.8.: What type of data to you analyse?

<u>Type</u>	<u>Count</u>
Gene panels	12
Clinical exomes	4
Whole exomes	2
Whole genomes	0
Other	1

Tab. D.9.: What format do you use when importing raw data?

Format	Count
Raw sequencing data (FASTQ, FASTA, etc)	4
Aligned data (BAM, SAM)	10
Aligned data and variant files (BAM + VCF)	6

Tab. D.10.: Do you analyze germline or somatic mutations?

Format	Count
Germline	6
Somatic	6
Both	4
Other	0

Tab. D.11.: What other NGS data analysis software do you use?

Software
none
No
Exomiser, PhenIX
Exomiser, Alamut
NextSeq
NextGENe (Softgenetics)
Oncobench
Amplifyzer, selbst entwickelte Bioinformatikpipeline
Torrent suite Variant caller and MiSeq reporter
torrent suite variant caller, MiSeq reporter
SnEff, custom solutions
JSI sequence pilot

Tab. D.12.: What keywords would describe your work?

Keywords
rare diseases research, muscular dystrophy diagnostics
Rare disease, neuromuscular disorders, mental retardations, cancers, epilepsy, etc.
rare diseases research
rare disease research
cancer screening
cancer screening
cancer screening
Cancer screening, therapy prediction
Cancer diagnostic
Screening of tumor-suppressor gene in breast cancer patients
cancer target
hearing loss research
skeletal diseases research
epigenetics, bisulfite sequencing, imprinting, separation of parental alleles
Muscular dystrophy diagnosis

Tab. D.13.: It facilitates the analysis of complex data sets (DNaseq, RNAseq, DNA methylation), *14 answers*

Disagree	1	2	3	4	5	6	Agree
	0	1	0	1	5	8	

Tab. D.14.: It allows to save time compared to other software, *12 answers*

Disagree	1	2	3	4	5	6	Agree
	0	1	3	0	4	5	

Tab. D.15.: It is more user-friendly than its alternatives, *12 answers*

Disagree	1	2	3	4	5	6	Agree
	0	1	1	1	4	6	

Tab. D.16.: It reduces biologists/scientists dependence on bioinformaticians, *15 answers*

Disagree	1	2	3	4	5	6	Agree
	0	0	0	4	6	5	

Tab. D.17.: GensearchNGS is my main NGS data analysis software, *15 answers*

Disagree	1	2	3	4	5	6	Agree
	1	0	2	2	4	7	

Tab. D.18.: It works reasonably fast on older/slower computers, *10 answers*

Disagree	1	2	3	4	5	6	Agree
	0	0	3	3	2	3	

Tab. D.19.: It helps me to do diagnostics more efficiently, *13 answers*

Disagree	1	2	3	4	5	6	Agree
	0	0	0	0	9	5	

Tab. D.20.: It helps me to answer research questions more efficiently, *9 answers*

Disagree	1	2	3	4	5	6	Agree
	0	0	0	2	5	3	

Tab. D.21.: Most useful functionality in GensearchNGS

Feature
visualization of variants (not only tables)
Compare Variants Tool
Comparison of different patients (de-novo analysis, compound heterozygote, homozygote...) Filters (especially Phenotype)
Combined lists of several patients, patients filter, quick-links to other useful databases like ExAC, the possibility to colour mark variants
filtering for reads with x % methylation
Anomaly Scanner
possibility of local library of already analysed variants
listing of variants in accordance of our parameters - visualization and identification of region of interest, coverage
Die Unterscheidung der elterlichen Allele basierend auf einem heterozygoten SNP
annotation and safeguard of annotated mutations
split view for reads and AA sequences
It helps me to solve ambiguous cases
Alignment viewer
Die Angabe vieler Informationen über eine Variante auf einen Blick: zb maf, Qualität, coverage f/r, Verknüpfung mit alamut und anderen datenbanken - » erleichtert die Interpretation von varianten

Tab. D.22.: Missing or incomplete functionality

Feature
possible improvement of handling of the CNV tool
more detailed manual
The possibility to leave notes for specific variants.
CNV-Analysis
perhaps a more complete solution from the creation of a library (with clinical information, tumor content %, expected genes in accordance with pathology) to the report. each
Anomaly Scanner
search function to quickly find other samples with the same mutations
annotation for multiple consecutive nucleotide deletions
To be linked to more databases
Verknüpfung von Sanger Daten mit ngs Daten eines patienten, cnv detection, Datenbank aller bereits bekannten Varianten (das hat es aber schon, oder?)

Tab. D.23.: Feedback

Feedback
best NGS analysis software I ever used; biggest advantage: quick help and very fast implementation of improvements in case of problems
Every Update makes GensearchNGS more powerful. Continue the development of Gensearch!!!
more detailed handbook of ALL functions
GensearchNGS hat mir sehr geholfen meine Daten zum ersten Mal anschauen zu können als wir noch keine anderen bioinformatischen Optionen entwickelt hatten, aber mittlerweile haben wir für unsere Anwendung und Fragestellung bessere Lösungen gefunden
very useful to determine variants and edit reports
So far I am satisfied. If I have troubles or I need something new, it is immediately done.
Ich arbeite seit 3 Monaten mit JSI seq Pilot und vermisse immernoch manchmal die Funktionen von GensearchNGS :)

E. Downloads

Here we list the different locations at which the software developed as part of this thesis can be downloaded.

GensearchNGS 6 : <http://www.phenosystems.com/www/index.php/products/gensearchngs>

GNATY 7.3, 7.2: <https://gnaty.phenosystems.com>

POP-Java 8: <https://github.com/pop-team/pop-java>

F. Declaration of Authorship

Ehrenwörtliche Erklärung

Hiermit erkläre ich an Eides statt, die Dissertation **Reducing the complexity of OMICS data analysis** eigenständig, d.h. insbesondere selbständig und ohne Hilfe eines kommerziellen Promotionsberaters, angefertigt und keine anderen als die von mir angegebenen Quellen und Hilfsmittel verwendet zu haben.

Ich erkläre außerdem, dass die Dissertation weder in gleicher noch in ähnlicher Form bereits in einem anderen Prüfungsverfahren vorgelegen hat.

Affidavit

I hereby confirm that my thesis entitled **Reducing the complexity of OMICS data analysis** is the result of my own work. I did not receive any help or support from commercial consultants. All sources and / or materials applied are listed and specified in the thesis.

Furthermore, I confirm that this thesis has not yet been submitted as part of another examination process neither in identical nor in similar form.

Fribourg, den

Beat Wolf