universität wien

BACHELORARBEIT / BACHELOR'S THESIS

Titel der Bachelorarbeit / Title of the Bachelor's Thesis

„Useable Highperformance Computing in Bioinformatics
On the example of next-generation sequencing methods“

verfasst von / submitted by

Kevin Sidak

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Bachelor of Science (BSc)

Wien, 2019 / Vienna, 2019

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	UA 033 521
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Bachelor Computer Science
Betreut von / Supervisor:	Univ. Prof. Torsten Möller,PhD

Contents

Abstract

In this paper I present different strategies and novel approaches to combat the challenges posed by the growing amount of data with the focus on performance and usability. I analyze different widely used tools in the next generation sequencing sector under the aspect of which tools are used and how they are integrated in the workflow of different researchers at the Vienna Bio Center as well as if these solutions are still considered state of the art. I also present new approaches like databases and map-reduce frameworks to make existing pipelines in molecular biology at Vienna Bio Center more efficient. As a result I provide guidelines for developing tools in the field of molecular biology in terms of usability.

1 Acknowledgements

I would like to thank Univ. Prof. Torsten Möller, PhD for his supervision and for pushing me to my boundaries and beyond without this constant motivation I wouldn't have made the same experiences in sciences. I would also like to thank Christoph Langer for his support and supervision at the Vienna Bio Center and the Gerlich Lab in particular.
I would also like to thank my colleagues at the University of Vienna. Especially my colleagues at the research group Visualization and Data Analysis and the Data Science research platform, especially Sebastian Ratzenböck.
Last but not least I want to thank all of my friends and family who supported me during the journey of this thesis. Notably my friends who accompanied me since the first semester: David Fuchsteiner, Lorenz Kummer, Thomas Fenz. Special thanks also to Verena Prantl who supported me not only with her friendship but also during the writing process and proof read the whole thesis.

2 Motivation

Working with the Vienna Bio Center(VBC) and the Institute of Molecular Biotechnology(IMBA) revealed a considerable demand of high quality and well performing software solutions for the preparation, analysis and storing of data related to bioinformatics, especially in the field of sequencing. Many problems in the area of molecular biology depend on how fast and efficient data is analyzed and filtered. These challenges are not well addressed by current software solutions and this fact is yet to be removed or at least improved. Searching for high-performance solutions (HPC) for next-generation sequencing (NGS) methods that are usable by computer affine molecular biologists proved to be a tremendous challenge.

The data generated in this field skyrocketed with the inventions of new parallel sequencing methods, the so-called next-generation sequencing methods. The current extraordinary long runtimes of the used tools are not acceptable. Scientists who are working in this kind of sector are in need of tools which have an easy learning curve and which they can use by themselves without losing valuable time on setup and usage.

The challenge grows continuously due to the dramatically increasing amount of data that is produced by current NGS methods and that needs to be analyzed. The previous solutions were not designed for this challenge and are the bottleneck of sequencing based approaches. They are slowing down research drastically in an increasingly expensive and competitive field.

Tools and software libraries do not implement the newest solutions that would be available in the field of high performance and scientific computing. Algorithms are not up-to-date and often not described properly in order to ensure fast qualitative research and, additionally, are black boxes.

This demand cannot be solved by bioinformaticians, molecular biologists, and computer scientists alone. They need to work together in order to produce best-practice qualitative tools that meet the current demand and can be used by researchers without prior training or extensive help or expert level HPC knowledge.

Although there were numerous high-performance tools released in this sector in recent years, they are not used by the scientists working in the Vienna Bio Center. Analyzing and testing libraries that are used on a daily basis by researchers revealed the lack of support and maintenance of many tools as well as practically non-existing documentation. Overall, many of the given tools are simply not usable.

To solve this problem, I first performed a survey to understand the requirements and problems of scientists as well as in what way tools are used and in which form data analysis is needed in general. Another important factor were the knowledge and abilities in computer science of the average scientist within this field. This last aspect is so important because without such a basic analysis, tools are not able to meet the requirements of the researchers.

In Chapter 3 I determined the requirements of the researchers in molecular biology with a survey and an interview on how tools are used and what interactions

are required. The questionnaire was sent out to all people currently working on the computing clusters of VBC. This survey gave me the insight I needed to identify the current state of software tools and how they are used, additionally to the fact, which abilities and what kind of knowledge are generally available. During the survey, it also became clear that available high-performance software is hardly used.
In chapters 3.2, 6 and 7 I concluded what a usable solution would look like and what the key components are, that it would require and why current solutions do not meet these requirements.
In addition, I found out that the tools used could easily be improved by using state of the art technology, if it was made accessible.
This explanation is shown very clearly in chapter 6, where I used different approaches to solve the demultiplexing problem, including clean and compact python scripts as well as databases. The speedup achieved was nearly a factor of 20 and actually saves two days of a four-day pipeline.
I analyzed the data in Chapter 4 and 5 and was looking for alternative approaches dealing with this massive amount of information. I was able to identify that databases and Spark frameworks would be viable solutions for this big data problem.
In chapter 7 I implemented Apache Spark into an existing pipeline to improve the runtime significantly. According to the survey, technologies like Apache Spark are currently not used in any pipeline, although Apache Spark promises a significant performance increase. This inability to use such frameworks is further discussed, and a solution is proposed in chapter 8.
Within this thesis I provide the first tool that allows simple integration of previous Spark frameworks into everyday used and most common pipelines (Nextflow). This solution is easily deployed on existing HPC frameworks without admin privileges and stable due to fully containerized solutions.
As mentioned above I showed that the four days, that it takes the previous piplines, can be shortened by two days, simply by fast demultiplexing for everybody in the institute. Further the alignment step in most NGS pipelines can be easily replaced.
In chapters 3 and 8 I discuss the new scope of the project, with a focus on usability rather than raw performance, and why the focus changed in the first place.
Finally in chapter 9 I identified what kind of solutions will be needed in the nearer future and stated some interesting follow up projects, to ensure qualitative research and improve the overall performance.
Without such analysis of the use cases and usage behavior as well as establishing guidelines and best practices, novel research in molecular biology is not possible and will become a matter of trust rather than facts. This is even noticeable in high ranking journals, like Nature, as shown later on in section 4.2.1.

3 Molecular biology background

This chapter is a short overview of the molecular biology background to understand the origin of the data and the challenges in this field. A detailed explanation can be found in the Appendix.

3.1 PCR - Polymerase Chain Reaction

The Polymerase Chain Reaction (PCR) is nowadays a widely used method to amplify specific DNA subsequences and to create millions of copies of these sequences. It was published in 1986 in Cold Spring Harbor Symposia on quantitative biology [35]. PCR enables us to amplify DNA extremely fast without having to multiply the DNA by growing bacteria. This is the reason why it is one of the most critical discoveries in genomics, and is the foundation for accurate and fast sequencing today. For example, it is used by Illumina [5], and it is also used in the Sanger method [41].

3.2 Sanger Method

The Sanger Method or Sanger Sequencing, published in 1977 by Sanger et al. [41], was a first generation sequencing method and builds the foundation for the next generation sequencing methods and is still used today. It was confirmed with an accurate whole-genome sequencing of the bacteriophage σX174 and was the standard sequencing method until 2005. By terminating the polymerase at specific bases and marking them, a reverse read of the target sequence can be obtained.

3.3 Illumina Method

The Illumina Sequencing Method or Illumina Dye Sequencing for accurate sequencing is one of the next-generation sequencing methods. It was first published in 2008 in one of the world's leading multidisciplinary science journals, called Nature [5]. In comparison to the previous mentioned Sanger Method, it is capable of less expensive, in terms of material and time, and more accurate reads. This is mostly achieved by reading multiple smaller sequences with about 200-600 pairs length at the same time and, in fact, the Illumina Method is parallelizing the sequencing process. In the appendix I have described the individual steps of this fascinating and highly elegant method.
The results are now hundreds of millions of forward- and reverse-reads, tagged with an index which needs to be sorted by their respective index. This process is called demultiplexing, after that these relatively small reads need to be aligned again.
Now it is clearer to see why usable tools for data analysis and data analysis in general are becoming one of the biggest challenges in Next Generation Sequencing, the amount of data produced through sequencing grows rapidly between the different generations. The most significant factors here are the reduced cost

and time necessary to sequence bigger DNA sequences. For example, in 2004, it took several years and about $300 million to sequence the human genome. Nowadays whole human genome sequencing can be done for $1000 and in weeks, which is a fraction of the cost, money and time-wise [38].

3.4 HiC

HiC is a method for structural analysis of a genome. It was first published in 2009 by Lieberman-Aiden et al. [30]. To get a better understanding of DNA sections close to each other, the HiC method uses ligation on cross-linked fragments. After this, the DNA gets sheared and only the pieces that were ligated are pulled down by Biotin. These remaining sequences that were physically close together can then be sequenced. The analyzing steps are explained in the HiC Pipeline chapter.

An extensive review of the analysis of HiC Data and the proposal of guidelines has been done by Lajoie et al. [25].

4 User Studies about tools used at the Vienna Bio Center

At first, the topic of this thesis had no focus on usability, but this changed quickly after working on the demultiplexing problem. The first step to understand this runtime problem entirely was to talk to an expert at the facility who uses the demultiplexing tool on a weekly bases. He explained that nobody up to now took a look at the source code of the illumina2bam [49] tool, but also admitted it would be a benefit if the process wouldn't take on average two days. After looking at different file formats and starting to work on a prototype in python, I noticed that there was a step prior to the demultiplexing in the header of the files. To be exact, I discovered that the sequencing facility used the bcl2fastq2 [18] on every single run. I discovered that this tool was already able to solve the demultiplexing problem by itself! This fact inevitably arose the next question: If there are better tools already why is nobody using them? To get a possible answer to this question and also a better insight into the tools, that are used at the Vienna Bio Center, I conducted a survey, did interviews about which software-tools are used and why, as well as general practices on developing and using these tools.

4.1 Quantitative Questionnaire

The get a general overview over the tools used and the computer sciences abilities available, I conducted a survey in form of a simple questionair which can be found in the appendix [23]. Another goal was to establish an answer to what is usable by the researchers in the molecular biology field working with next generation sequencing. The people that were chosen for this survey are all working on the computing clusters of the VBC, therefore use software tools

regularly.
In the following are the key insights I got from the survey:

- **What is your position in your research group/department? (single choice)**

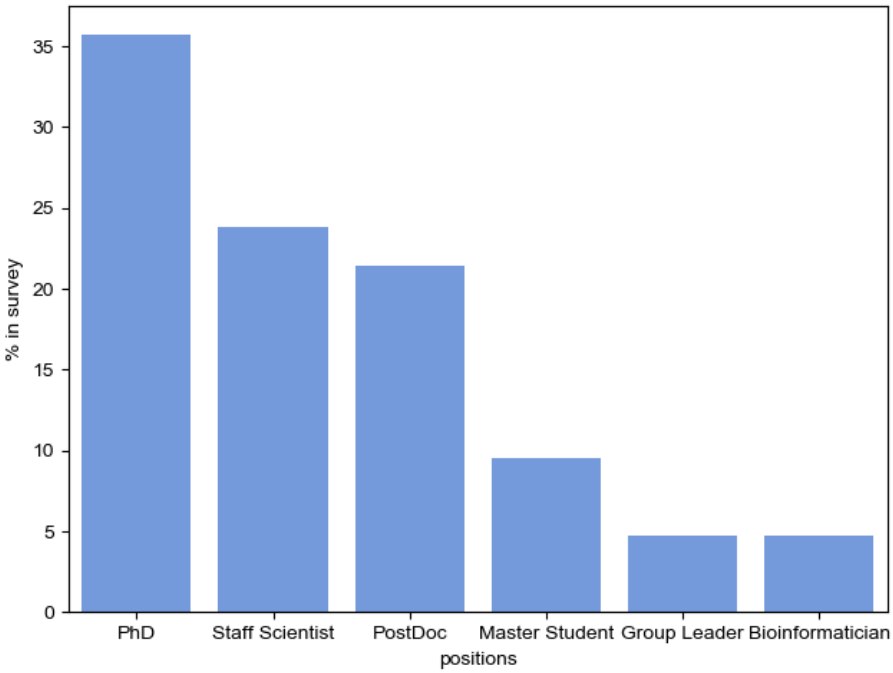

Figure 1: Positions in the survey

I got 42 individual replies from the study from different positions at the VBC: 35.7 % came from Ph.D. students, which was expected since they represent the majority of employees (See figure 1) .

- **From which journals do you get information about bioinformatics tools? (multiple choice)**

  As seen in figure 2, journals named the most where Oxford Bioinformatics and BMC Bioinformatics as well as in some cases Nature Methods. The interesting thing about these answers was the fact, that no Computer Science or Big Data Journal was named, although some of the used tools like Stream Aligner by Rathee and Kashyap [40] or a review paper about "Scalable Bioinformatics Pipelines" by Fjukstad and Bongo [12] where published in the Journal of Big Data and Data Science and Engineering, not in any bioinformatics journal.

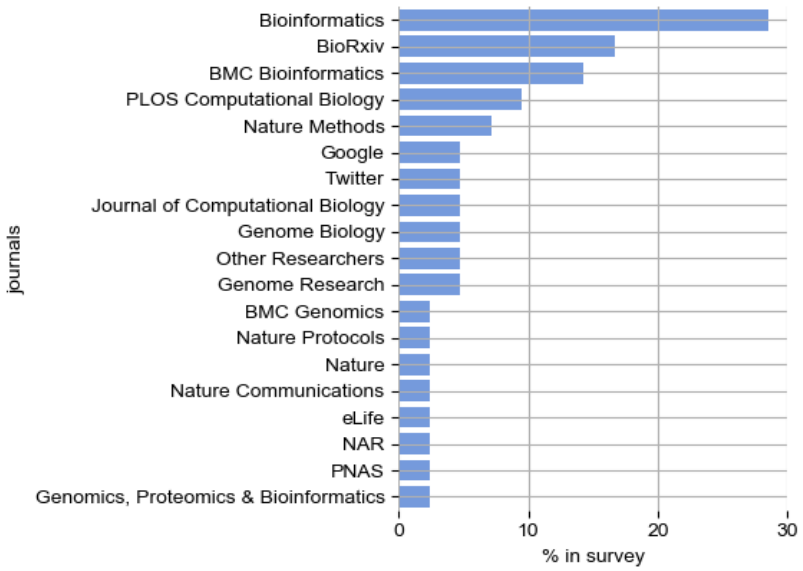

Figure 2: Journals in the survey

- **How do you choose your bioinformatics tools? (multiple choice)**

  As seen in figure 3, 88.1 % answered that they look up the tools they need with Google, and 81 % say they also use tools recommended by colleagues. Only 42.9 % use journals as a source for their tools, which makes journals not the main source for finding new or better tools. Also, 54.8 % of participants stated, that they are modifying existing tools.

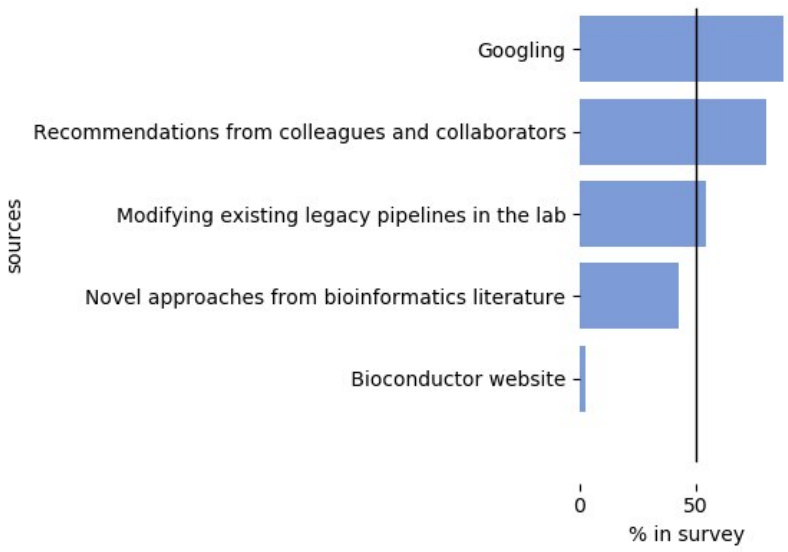

Figure 3: How tools are chosen in the survey

- **User or Developer? (single choice)**

  The majority of answers came from the tool/pipeline users with 50 % as seen in figure 4, this participant ratio also had an effect on the programming and usability section of the questionnaire.

- **How much is your time split between bioinformatics and wet-lab work? (single choice)**

  Here 40.5 % say they only do computational work, suggesting that most of the participants where bioinformaticians.

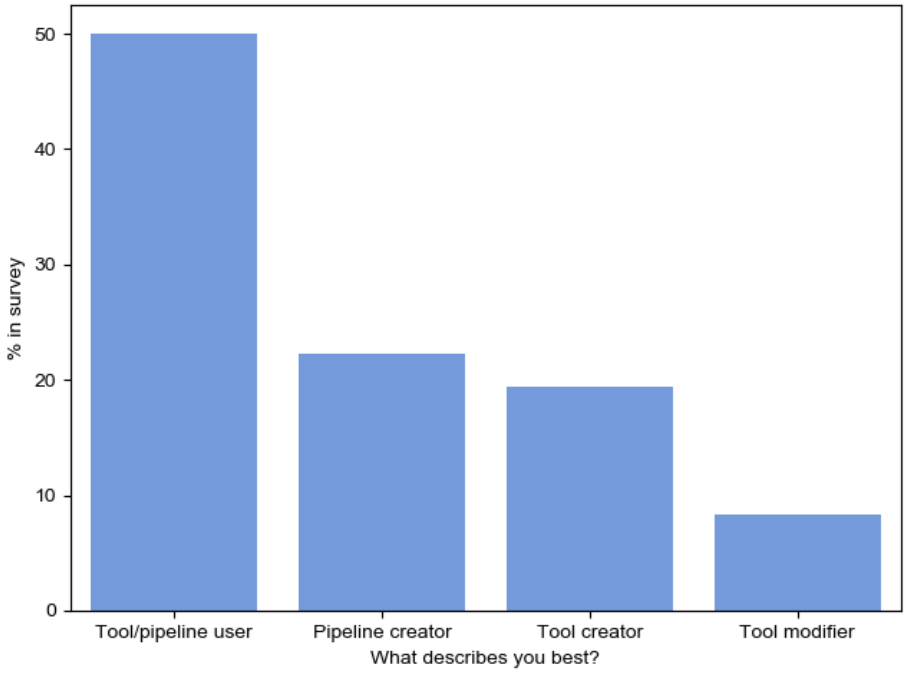

Figure 4: Ratio Users and Developers in the survey

- **Programming section (multiple choice)**

  The participants rated their own programming skills (single choice), as displayed in figure 5, which is curious since 40.5 % said they only do computational work. This may be explained in the way people learned programming.

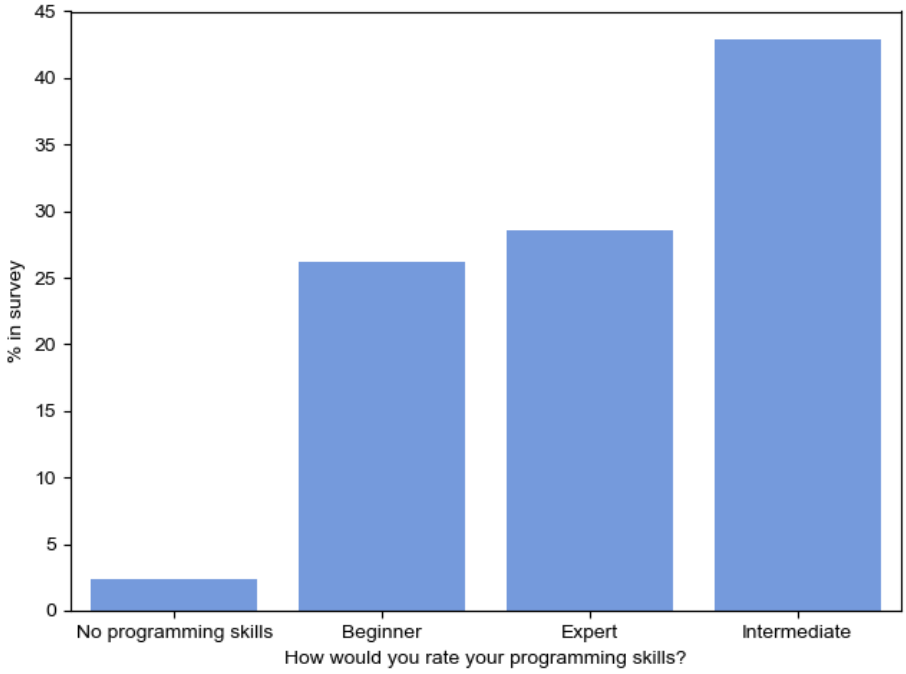

Figure 5: Programming skills in the survey

- **Programming languages and concepts (multiple choice)**

  By far, the most used languages are R (82.9 %) and Python (68.3%) as seen in figure 6. Regarding things like databases, parallel computing or virtual environments, it came out that they are known by over half of the participants (between 54-60%) - of which only 10.5 % know the map and reduce concept used by Apache Spark. Although most people stated for example that they only use virtual environments in Python in the form of Conda. This can be seen in figure 7.

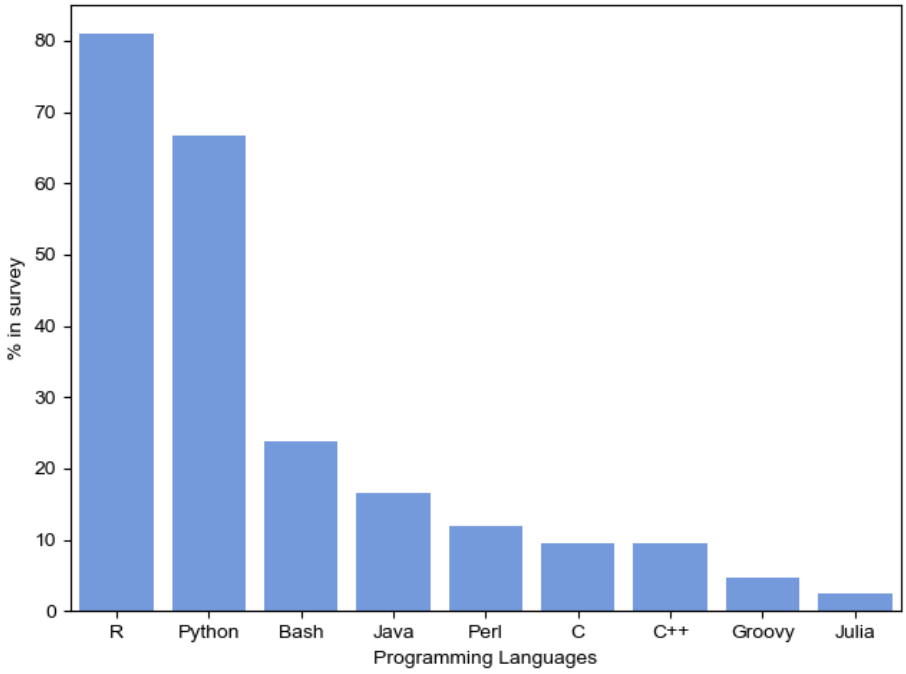

Figure 6: Programming languages in the survey

- **Who is the target user group at the VBC?**
  Only in 31.5 % of the cases software gets published outside of the VBC.

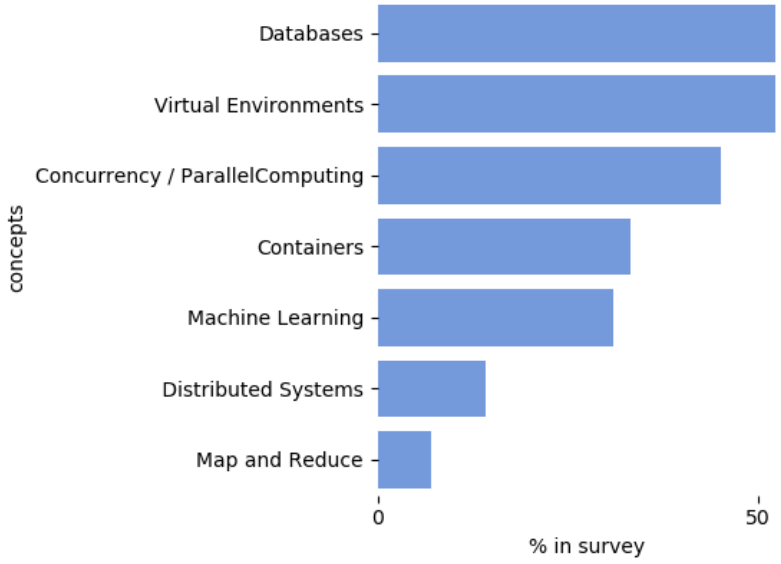

Figure 7: Concepts known in the survey

- **Which interface or interaction is required?**

  The most needed interface is the command line followed by only including software libraries without any interface. Graphical user interfaces on the other side are hardly needed as seen in figure 8.

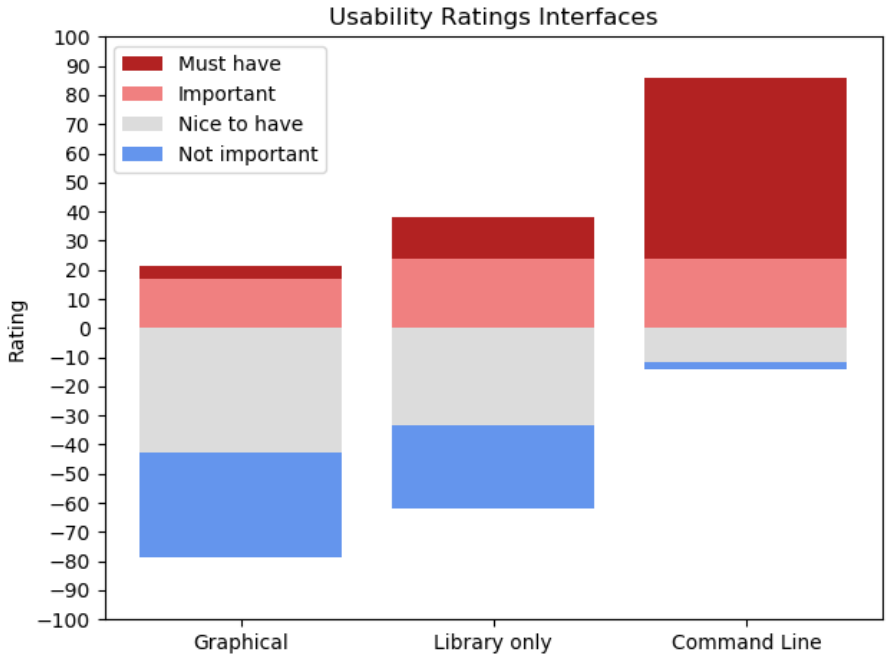

Figure 8: Which interfaces are preferred in the survey

The primarily needed output formats are standard output in the terminal and visualization in general as seen in figure 9.

- **In which form do you want your results?**

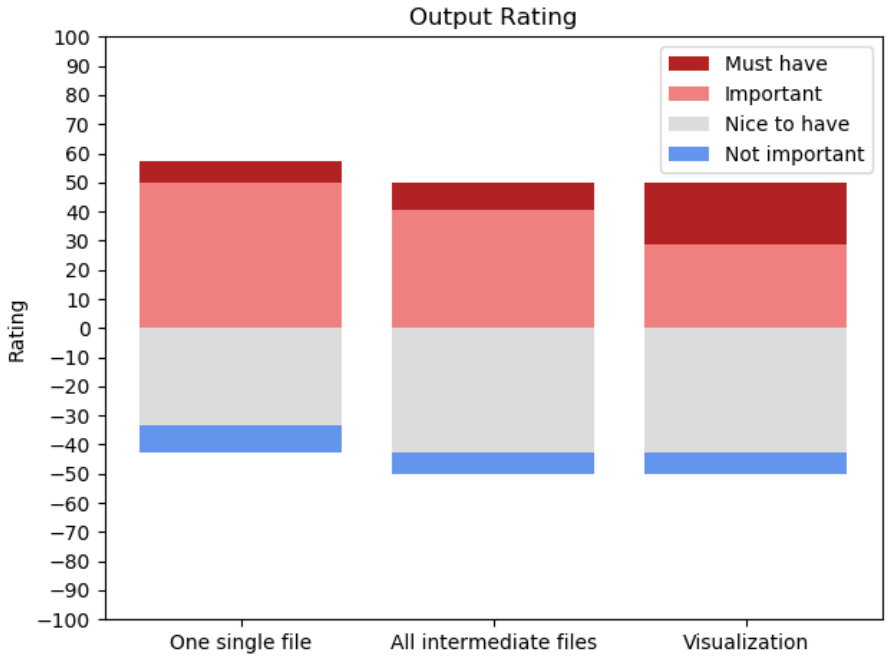

Figure 9: Output format

- **How do you choose between multiple tool options?**

  As seen in figure 10 on one hand key features of a tool for the participants are:

  - Detailed documentation
  - Known programming language
  - Opensource
  - Ability to adapt and compatibility with existing tools

  On the other hand not relevant are:

  - Video demos
  - Last Update date
  - Runtime
  - Automation
  - Ability to adapt and compatibility with existing tools

  This suggests that the focus of the analysis is shifting towards quality instead of quantity.

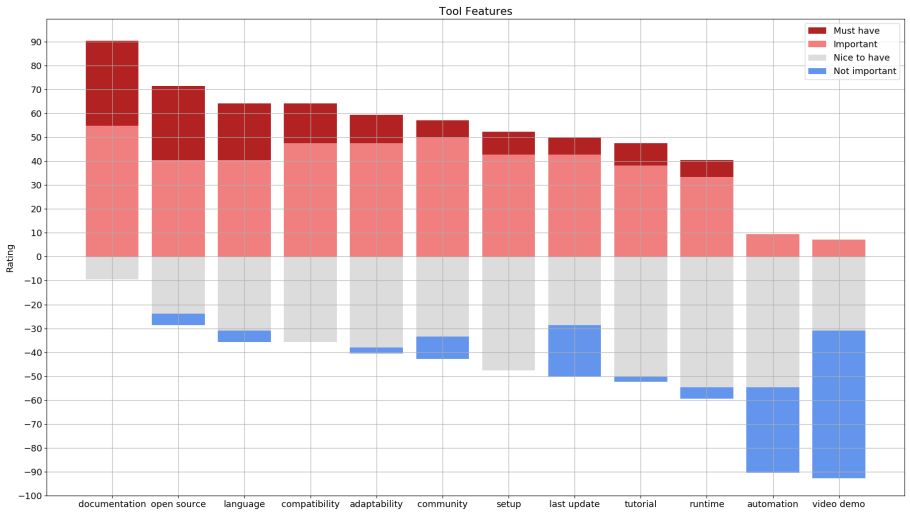

Figure 10: Features rated in the survey

- **What is the greatest bottleneck in your analysis workflow?**

  The essential issues in current pipelines were: the lack of proper documentation, their own coding skills and their available time are not sufficient to modify or even create newer and better performing tools in terms of quality and computing performance.

- **What is the one thing that you think could advance the field of bioinformatics in the next year?**

  The most anticipated topics were better documentation and guidelines or best practices for the field of bioinformatics.

- **Conclusion**

  The decisive insight I got from this survey was, that for the most users performance is not as important as the ability to understand what a tool does, in the form of documentation or as part of the output. Most users lack the knowledge and time to develop software themselves so instead, they are just trying to modify existing software and patching together multiple individual scripts.

4.2 Qualitative Interview

4.2.1 Molecular Biology Ph.D.

The Interviewee is currently working as a Ph.D. at the VBC, IMBA in the Gerlich Group, he is currently working on the HiC pipeline as well as general

analysis on NGS data.
In the interview additionally to the questions and topics in the questionnaire more detailed questions were asked about different tools to solve the demultiplexing problem as well as the HiC pipeline and the currently used software tools in the NGS sector in general.
These are the following insights and concerns he revealed in the interview:

- For him, one of the most important things about a tool are how difficult it is to install and how easy is it to use. He added that this is one reason why tools are not up to date since it would require additional time to switch to another tool with the possibility that this tool would not work in the end. Essentially the risk that he wastes precious time on a tool that is faster but might not work is too high. Because of the poor documentation and absence of maintenance in this field this setup time is not calculatable.

- Another direct result of no documentation is that often it is not clear what a tool does in detail. According to him most tools have generic names and the researchers have to look into the source code to find out if a tool has the functions and methods that they need.

- Performance is only an issue if it affects his general workflow and has a significant impact on the time it takes to analyze one experiment. Usability is of much greater importance since he has not enough background knowledge to solve problems occurring during setup and runtime of tools efficiently and by himself.

- If a tool provides a visualization of the results or not is not essential for him. In most cases it is actually annoying for him since it makes the tool more complicated than it has to be. If he needs some form of visualization, he plots it himself and only wants the raw data from the tool.

- Regarding pipelining his toolchain, Michael had the problem that at the beginning a lot of the steps changed continuously and only for the last months he had a stable workflow and order of tools with which he prepared his data. He does not have the time yet to look into automated pipelines but would be interested in using them, if they are easily integratable and usable. Another important thing would be that intermediate results are always accessible and that the system automatically detects errors in the provided files or configurations and informs the user. These sanity checks should be definable by the user as a ruleset. Visualization and plots would be a feasible representation for this quality control mechanics.

- A tool that would help him decide which tools to use and building the pipeline would be an acceptable solution for him. The portability, for example, in the form of containers would be crucial for him, since he does not want to be dependent on the IT department for every step in the pipeline or the tool itself.

- He stated that the current state of usability in bioinformatics tools is horrible. According to him, he had to take a look at the source code of more than 20% of the tools in order to figure out how to use them and what they do. The lack of knowledge about and control of how a tool processes the data leads to uncertainty that unfortunately, because of limited time has to be accepted as is. According to him this is a general problem in the field, since the results in publications can no longer be verified by the reviewer nor reproduced by other scientists. An example of this mentioned by him was the paper "Repair of double-strand breaks induced by CRISPR–Cas9 leads to large deletions and complex rearrangements" published in Nature Biotechnology 2018 by Kosicki et al. [24]. The analysis pipeline used in this paper was: "[...]All downstream analysis was performed using custom R (v 3.3.2) and bash scripts and visualized with ggplot2 package.[...]" [24]. The mentioned scripts were never published. For him, having better documentation and usability is an essential step towards higher standards regarding reproducibility in research.

4.3 What is Usability in this field?

As already shown the main problem is the lack of documentation and simplicity in order to setup and use better performing tools. Since not every lab has a computer scientist available, the tools themselves have to adapt. Every tool used regularly has to be maintained and updated to ensure compatibility within evolving environments. This is evidence that our computational tools should have a strong component of explainability.

5 Guidelines for tools in bioinformatics

In this section, I will elaborate on guidelines I worked out from the survey and the interview. These guidelines are not fixed rules and have to be modified for each specific case. They should help with the processes of making tools more accessible by researchers in the field of molecular biology without any training or prior computer science knowledge.

- Provenance: This is probably the most critical rule for tools in bioinformatics since it is currently the most significant problem in this domain. As shown in both user studies currently, it is not easy or impossible to reconstruct and reproduce the analytics steps that are used for most papers in molecular biology. It should be clear what exactly happened to the data when a tool is used and have well-defined inputs and outputs. It is also necessary to document the intermediate steps to achieve this goal. Every step and algorithm used has to be documented so that it is accessible and understandable by the user.

- Documentation: As mentioned, the documentation of the tool is essential. I would argue that this is the main reason for the problem of no

provenance. The documentation should at least include the following:

  - All available options and parameters
  - Description of all data transformations and analyses that are done by the tool.
  - All input and output formats that are available.
  - Installation or compile guide if necessary
  - List of any dependencies needed
  - Recommended best practices
  - Example use cases

  This documentation should be made understandable for users with little to no computer science background so they can use them on their own. Similar guidlines are proposed by Mangul et al. in 2019 [31] and Forward et al. in 2002 [13].

- Maintenance and maintainability: This is probably the most challenging guideline to fulfill. Providing continuous maintenance for a tool is were most tools fail and get outdated. It requires either a team of developers or an institute who takes care of the project or a community that is big and dedicated enough to continue working on the project even when the initial developers leave. For this to work, the maintainability of the project is essential. This boils down into using known languages and technologies as well as documenting the code and enforcing coding best practices in general. A workaround for this could be the usage of containers and virtual environments.

- Portability: To ensure that the tool is used in almost all environments and setups, someone should consider either providing enough setup options for different systems or pre-configured containers. In a time where containerizing software like Docker and Singularity is standard on every major cluster and system, it seems reasonable to build a container around the tool, especially when uncommon libraries or languages are required.

6 Data Characterization

In order to be able to develop tools in the molecular biology domain, I had to characterize the data that is commonly employed, by using the data characterization methods described by Munzner [36].
The data always consists of independent string sets which are encoded by a defined set of characters. Each of these string sets stands for itself and itself alone, which makes it possible to analyze them in parallel without the need for data synchronization or the problem of race conditions.
The single characters used to encode the information in the different file formats is limited and in most cases consists of an encoding for the bases adenine,

cytosine, guanine, thymine, and uracil or a multiple of them. The different files as well as the allowed characters are discussed further in the following chapter.

6.1 Description of File Formats

In order to develop better analyzing-tools in the field of next-generation sequencing, I had to take a closer look on how the data is stored and which files are used in the domain, to guarantee the compatibility with other domain-specific tools.

6.1.1 SAM - Sequence Alignment Map

The SAM format as described by Li et al. in 2009 [28] is a tab-separated text format, the main difference to regular tab-separated files is that this format consists of two parts: a header section, which is often more than one line and a body section, which contains the alignment information and general information about the read. The header is not mandatory, but the body consists of at least eleven mandatory fields. First, the different header fields and flags will be explained:

Header Each header has to start with an "@" followed by the header type, which is a two char code. In the files provided by the IMBA, there were only two tags used @HD and @PG.

- @HD: This is the starting sequence of the header and indicates that a header exists in the first place.
- @PG: PG is used to indicate which programs were used to process the file or to transform it.

Body The body has to consist of eleven mandatory columns, the most notable ones for this work are:

- SEQ: In the SEQ column, the sequence is stored, which is usually a string, containing A, C, T, G and N for Errors.
- QUAL: This represents the quality of the read.
- TAGS: The barcodes were just attached as general tags.

How to encode files in the SAM or BAM format is specified by the guidlines of the SAM/BAM Format Specification Working Group released in 2018 [46]. Unfortunately, working with IMBA revealed, that most libraries, tools or users are often ignoring them.

Tags and Barcodes As previously explained, IMBA uses the additional and optional TAGS for the barcodes. The SAM format allows three different kinds of barcodes.

- Sample Barcodes: As used by IMBA, these barcodes help to identify which sequence belongs to which library or, in other words, which sequence belongs to which experiment, specified in the sample sheet or barcode table. This information is marked with the tag "BC:Z" and it also has a mark that states the quality, namely "QT:Z".
- Cell Barcodes: Very similar to the Sample Barcodes, but they can only help identifying, if two sequences come from different cells and only if one is using more than one cell in a single cell analysis.
- UMI: This is only used if there are two similar sequences from different lines, and it is unsure, if this is due to a PCR artifact or not.

6.1.2 BAM - Binary Alignment Map

The BAM file format as described by Li et al. in 2009 [28] and SAM/BAM Format Specification Working Group in 2019 [47] is a compressed version of the previously mentioned SAM format. It uses its own block compression algorithm called "BGZF" in the background to compress the information of the SAM file without any loss of information. Since BGZF compresses the information in uneven blocks, random access is not possible without creating an index for the BAM file. For indexing, the BAM file has to be sorted by the reference IDs and then these IDs are referenced with their corresponding block sizes in the BAM index file.

6.1.3 FASTQ

The FASTQ file format was developed by Jim Mullikin at the Wellcome Trust Sanger Institute but was never officially documented as described in [8], the nearest thing to some documentation is the Maq documentation [15] written by Heng Li at the Wellcome Sanger Institute. Solexa and Illumina have developed their own versions of the FASTQ file format, with the change being the altered quality score.

The FASTQ format is a simple multiline format that can only hold the quality score and the sequence itself.

Formation

1. Block Description: This line with a leading @ holds the description information for each sequence.
2. Sequence: Means the whole sequence
3. Separation: + is used as a separator between sequence and quality

4. Quality: Meaning the quality information for the previous sequence

These four lines combined are one block and represent the information on one sequence. If more information is needed, for example, the barcodes of the different sequences before demultiplexing, they have to be saved in their respective own FASTQ file and can than only be matched by line number.

6.1.4 Analysis of Fileformats

As a reference, the same sequencing run of a Novaseq machine is used for comparing the different file formats. This run holds two times (forward and backward read) 540 million sequences where each sequence has a length of 250 bases in addition to their barcodes as information, which results into a SAM file with 610 GB. I compared SAM and FASTQ compressed with the compression levels one to nine as seen in figure 11. I also included BAM since it is the standard compressed format used for SAM files.

6.1.5 Size

Regarding the file size, figure 11 shows the relation between the filesizes and the compression algorithm and level. The difference between FASTQ and SAM is small since the main difference is only the number of rows and separators. In general, SAM files are smaller.

- **LZ4:** LZ4 seems to have its compression limit at about 140 GB which is a compression rate of 77 %.
- **GZIP:** GZIPs limit seems with all compression rates around 112GB this is equal to 81.6 % compression.
- **BZIP2:** BZIP has up to now the best compression rate with 92 GB as best compression rate (84.9%), but even level one compression is with 84 % better than level nine of LZ4 and GZIP.
- **BAM:** BAM is very similar to GZIP level 5 since the internal compression algorithms are very alike. This results in a BAM file with 116 GB.

6.1.6 Performance

Performance is the second important factor regarding which file format to choose and which compression algorithm.

- **LZ4:** LZ4's level one compression is impressively fast; it takes only 65 minutes to compress the whole SAM file. However, with ascending compression level, it gets slower in time, to the point where bzip2 is faster while maintaining a better compression rate.
- **GZIP:** Overall GZIP had the worst performance, although the first five levels are faster than any BZIP2 compression level, the compression rate is worse as a trade off.

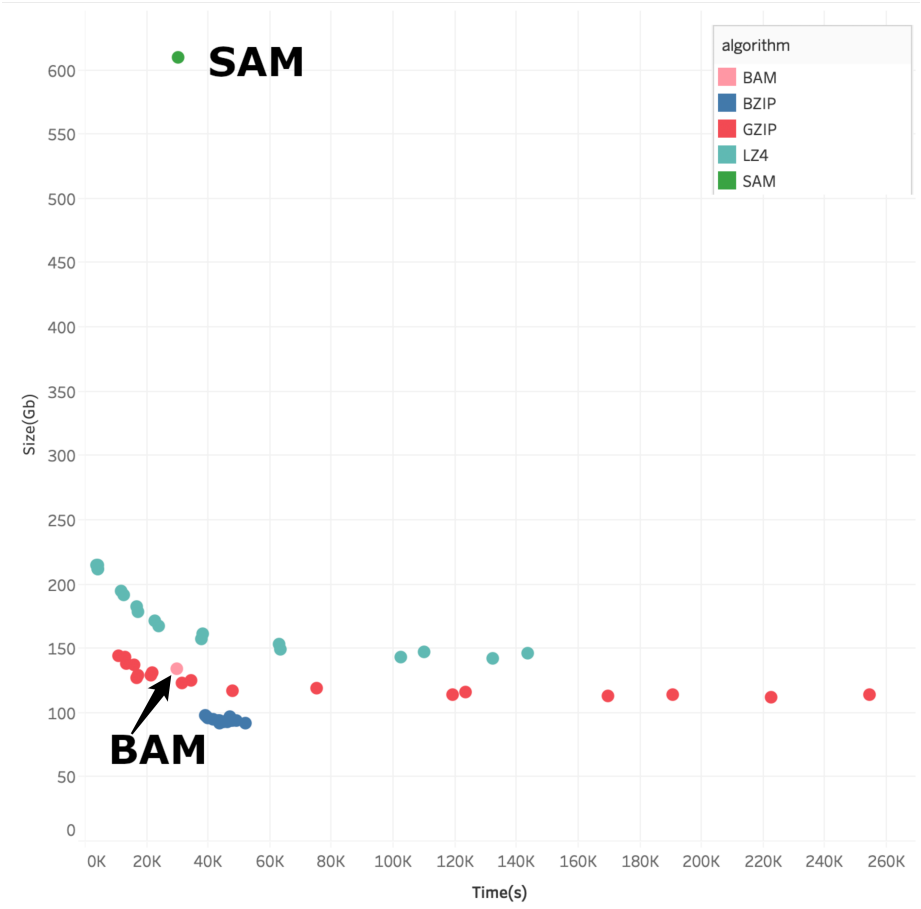

Figure 11: Comparison of fileformats regarding time to compress and size afterwards. Calculated on 8 cores of a Intel® Xeon® Gold 6138 Processor

- **BZIP2:** BZIP2 is not as fast as LZ4 or GZIP in the first five compression levels but has overall the best performance/size ratio. All levels of compression of BZIP2 ran between ten and twelve hours.
- **BAM:** Since BAM uses GZIP 5 internally, it is relatively close to GZIP performance-wise.

6.1.7 Usability and Conclusion

There is no clear winner regarding compression speed and level. Although BAM is widely used, its lack of parallel decompression without building an index first is a real downside. LZ4 has no high compression rate but can be fast and paral-

lel decompressed. The best trade off between speed and size has BZIP2, which also can be parallel decompressed with pbzip2[20] , and this would make it an ideal candidate to be used as a resulting file format of bioinformatics tools or at least to store the files for documentation purposes.
Regarding uncompressed files there is a tradeoff between parallel reads and centralized information. SAM contains all pieces of information about one lane read including the barcodes which makes it easy to work with in terms of simplicity, also parsing a SAM file is trivial since it just contains tab-separated values, where one row is one read. FASTQ has the downside that it is split up between multiple files if, for example, barcodes shall be included, but this attribute also enables parallel access more easily since multiple files can be read simultaneously in parallel.

A significant drawback is a multirow property, which means that one data entry is described not by multiple columns but by multiple rows. This posed a challenge in terms of writing a parallel parser. Most simple file parsers expect a single line format with a separator, which is one reason that FASTQ files are not as suitable as SAM files when used with Apache Spark since it expects a single-line format. In case of databases FASTQ requires another join between sequences and barcodes to retrieve all the information needed.

7 The demultiplexing problem

In this chapter, I will take a closer look at the challenge that occurs with the multiplexing of different experiments in the Illumina sequencing machines mentioned in the Illumina method chapter. I will analyze the existing tools to handle this process as well as my approaches in terms of performance and usability, considering the survey and interview conducted at VBC.

7.1 Reference Implementations

The reference in terms of performance will be an extension to the Picard tools [7] called Illumina2bam [49] and the bcl2fastq2 tool [18]. Both tools are heavily used by the scientists working at VBC.

- Illumina2bam: Illumina2bam is a Java-based software collection for working with data concerning next generation sequencing, demultiplexing is only one use case of this tool collection. It uses the Levenshtein distance to sort the reads to the provided barcodes [49].

- bcl2fastq2: bcl2fastq2 is the tool provided by Illumina to convert the BCL files from their sequencing machines to FASTQ files. Since this tool requires the raw data from the machine, it is often not used by the scientists themselves but rather by the facility doing the sequencing. Usually, the BCL files are not given away by the facilities. Additionally, the demultiplexing step is only optional and is not always done by the facility or only

for a fee; the conversion to FASTQ is mandatory since almost no other program can read the BCL format [18].

7.2 Naive Python Implementation

My first approach for the demultiplexing problem described in the Illumina method section was to build a simple python program and try to optimize the I/O operations as well as the sorting strategy used. For that, I also took a look at the multithreading and multiprocessing approaches python provides.

7.2.1 Theory

Python is only an interpreted language that means it needs an interpreter to be read. For my approach, I used the commonly used CPython interpreter, which is written in C and can translate the python code to bytecode which is readable by the OS and CPU.
This additional step between python and the OS is also the reason why python uses a "Global Interpreter Lock" to avoid race conditions. This lock limits the number of threads that can execute a given python file at the same time. This is pythons strategy to cope with this problem and is not a general statement about interpreted languages. This lock can not be controlled by the user or the programmer and is more of a system and concept limitation than a design choice. To circumvent this, instead of threads processes are used. However allocating processes costs more time than spawning a thread. Another thing python standard implementations automatically do to optimize the locking, is to release the lock if the thread is blocked by an I/O operation [39].
I chose python because its a standard programming language in molecular biology as seen in the survey. Also, libraries like pysam [3], make it easy to access file formats like FASTQ, SAM, and BAM. Besides the big scientific community, Python code is also easy to read and modify by programming apprentices and has a simple syntax. Since the demultiplexing problem is mostly I/O bound, the limitation of the Global Interpreter Lock has only little impact on the runtime and overall performance.

I used two different algorithms to sort the barcodes to the right experiment.

- Mutations: I first generated mutations for all possible barcodes by substituting every char in the barcode with A, C, T, G, which produces a mutation of the given barcode every time. This generation is done n times whereas n is the number of errors the user wants to allow in every single read.

- Levenshtein distance: The Levenshtein distance was first described in Doklady Akademii Nauk in 1965 by Levenshtein, V. I. [26]. It compares two different strings and measures the distance of the two strings. The algorithm does this by considering insertions and substitutions of characters.It can be used for two strings that do not have the same length, which

makes it more fitting for demultiplexing than, for example, the Hamming distance which requires two strings of the same length.

7.2.2 Implementation

I decided to implement my approaches via python since it is known in the community and versatile. These are the key components of my implementation:

- Since it was the most requested interface, I provided a simple command-line interface with a manual. Each parameter can also be specified in a config file or via environment variables.
- I used the pysam library [3] to load the SAM file of the run and then extracted the sequence information as well as the barcode fields.
- The most crucial part of the program is the sorting algorithm and writer for the BAM files. As previously mentioned, I tried two different algorithms for sorting the sequences. I additionally included sanity checks. For example whether a barcode that was not specified by the user occurs in the run. Every target file gets its own file writer to avoid race conditions and also blocking between different writers.

7.3 Databases

In the paper "MapReduce and parallel DBMSs: Friends or foes?" by Michael Stonebraker [43], it was already shown that database systems can compete with Map and Reduce Frameworks like Apache Hadoop. Although this was before Spark was released, databases are still extremely efficient in storing and simple analysis tasks, without longer pipelines to make use of Sparks in-memory methods. Most Database systems even use compression algorithms so that the I/O waiting time matches the computation time for the decompression, which leads to a costless compression. Another essential point about databases is the excellent support of the major relational databases like MySQL, MariaDB, PostgreSQL, and also NoSQL Databases like MongoDB; I will explain the difference in the next chapters.

7.3.1 PostgreSQL

PostgreSQL is a relational database which complies with the ACID standard. It was first published by Stonebraker et al. in 1990 [44]. It is nowadays available on every major OS and is accessible through different client interfaces as well as via an API that is implementable in most programming languages used today [45]. One reason why I tested PostgreSQL as a possible data management solution, as well as a solution for the demultiplexing problem, was the ACID standard, which is defined as followed [17]:

- Atomicity: Atomicity means that transactions which are changes to the database are either entirely completed or the state of the database does not change. This prevents an undefined state of the database.

- Consistency: The database being consistent means that all data has to comply with the constraints and rules defined in the database.
- Isolation: Transactions are always isolated meaning that multiple transactions cannot affect one another if they are not completed yet, preventing undefined states.
- Durability: Following the principle of Atomicity once a transaction is fully completed in the database, this change is permanent and is guaranteed to be so even if the system shuts down immediately afterward.

The second main reason was the fact that PostgreSQL fully supports all primary functions of the SQL standard, which means that with this powerful query language, I was able to solve the demultiplexing problem within the database without the need of an external program or tool, besides the conversion to and from the domain-specific file formats.
Another thing to consider is where the database is within the CAP theorem, which was first presented by Eric Brewer at the Symposium on Principles of Distributed Computing (PODC) in 2002 [6]. It describes three attributes of distributed systems where every distributed system can at one point only fulfill two of these three properties. Depending on the use case one has to choose what properties are needed in a distributed system. It is defined as followed:

- Consistency: Similarly to the ACID Consistency, it guarantees that once a transaction is committed, the state of every component within the system of this data is the same.
- Availability: Every connection and request to the system is always handled in an appropriate time window.
- Partition tolerance: The tolerance to components failing within the system due to, for example, network problems or system errors, but still being able to handle incoming requests without data loss.

PostgreSQL and the system I was building are Consistent and Available but had no partition tolerance since this was not the main focus for my prototypes.

7.3.2 SQLite

D. Richard Hipp developed SQLite in 2000 [16]; it is also a conventional relation based database. The main difference being, that one instance of SQLite is one self-containing file instead of a server. That is also why most of the time, it is local and within the client application itself rather somewhere remote. It also lacks the ability of roles or users, meaning there is no access control, so anybody who has the file can access the data within. Usually database servers can define their own file system on the disk instead of using the default of the operation system. Since SQLite is only a file it cannot create its own file system on the disk therefor it has to optimize everything within one file. This is not as efficient as having a dedicated section on the disk.

7.3.3 MongoDB

MongoDB, where Mongo stands for humongous, was released in 2009 by MongoDB, Inc. and is still maintained by this company [33]. It is a document-based so-called NoSQL database but has full ACID supported transactions. MongoDB has no relations; instead, it uses a JSON document-based system of collections. Every collection can have multiple JSON Objects with different layouts. NoSQL, Not only SQL, means that MongoDB has no traditional SQL query language embedded but instead has its style consisting of filters and projections rather than join operations which are still supported but not commonly used.
I choose MongoDB as a NoSQL database because it is well optimized for big data problems, and the filter options are fitting for the task of demultiplexing. Another reason was that the sequences and barcodes can be easily represented as objects instead of using a relation.

7.3.4 Implementation

For the demultiplexing, via databases, I choose python again as a programming language. In this implementation, I also added multiprocessing support as well as locks to workload balance the different data streams to the databases. Here are the key components of the database implementation

- I started with a python module that takes care of handling the FASTQ files provided by the bcl2fastq2 tool. I needed to implement a parser since existing solutions couldn't handle separate FASTQ files for sequences and barcodes. This parser then combined the different FASTQ rows to single objects that could then be inserted.

- A database handler that established a connection to the desired database depending on the settings chosen in the config. It returns a generic connection object to avoid repetitive code segments for the different databases.

- A dispatcher that starts an insert job for every FASTQ file provided. The jobs are started as processes with the parser and the file as arguments as well as a reference on the lock to coordinate and balance the inserts into the database.

7.4 Performance

In this chapter, the different strategies for the demultiplexing problem are compared. There is a slight difference between bcl2fastq2, Illumina bam decoder and the database approaches since the primary workload was the insert in the database rather than the sorting, which is already done due to primary key and indices. In table 1 all measured performances are listed.

method	insert [min]	1 error [min]	2 errors [min]
bcl2fastq2	0	20	24
Illumina2bam	0	1945	2040
python mutations	0	468	3242
python levenshtein	0	148	122
SQLite	140	>1200	>1200
PostgreSQL	1341	2	2
MongoDB	397	20	28

Table 1: Performance of all approaches and existing tools.
method: this represents the program/method used for the demultiplexing;
1 error: demultiplexing allowing one error in the tag;
2 errors: demultiplexing allowing two errors in the tag

- bcl2fastq2: bcl2fastq2 is the clear winner in terms of runtime, which is not surprising since it is the only tool that can run on the native files produced by the sequencing machines. The files from the machine are small in size but numerous, so it is easier to implement parallel strategies on multiple files instead of one single big file.
- Illumina2bam: Since this tool also uses the Levenshtein distance as the measurement for demultiplexing, the timing between one and two errors in the string to match are not different. One issue with this tool was its usage of the RAM which skyrocketed, making it hard to predict if it will finish without crashing.
- python Levenshtein: The Levenshtein strategy was the best self-implemented prototype for demultiplexing with an acceptable runtime in case the BCL files are not provided
- python mutations: Unfortunately, this strategy grows exponentially with the errors allowed since it has to check against more strings.
- SQLite: Extremely fast insert, but didn't finish the join beyond twenty hours to do the real demultiplexing.
- PostgreSQL: High insertion time but the demultiplexing itself was done in two minutes, would be a good solution if more steps can be applied at the database.
- MongoDB: Good insertion time when compared to the PostgreSQL database. The join logic was more complicated to implement since joins are not commonly used in a NoSQL environment.

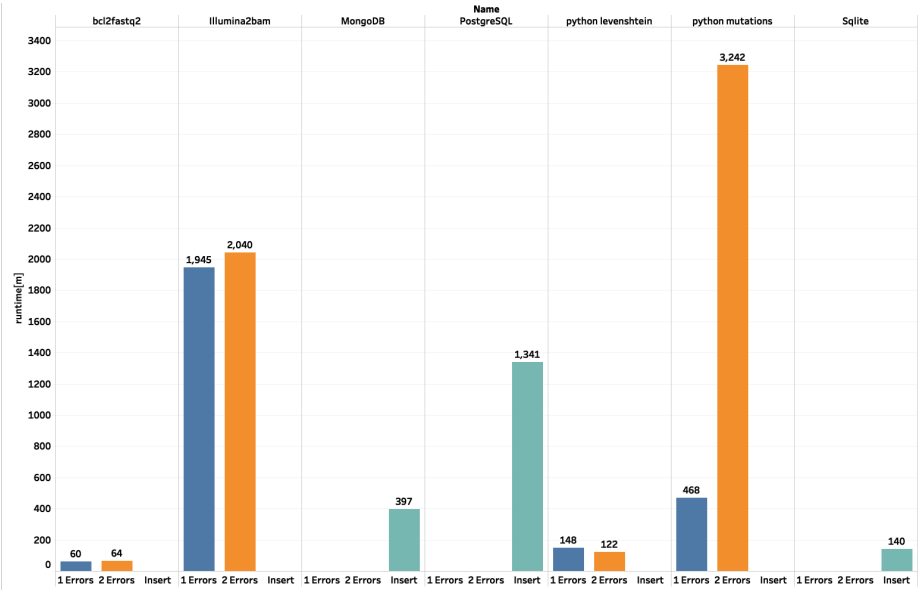

Figure 12: Performance of demultiplexing tools. The colors indicate the category. blue: 1 error; orange: 2 errors; cyan: insertion only

7.5 Usability

In terms of usability, there is a clear distinction to make between setup and execution, since for example, databases require a server as a setup but nothing additional if a python client program is used. The self-made python tools were developed considering the insights from the survey and the interview with the molecular biology Ph.D., who also tested the tools.

- bcl2fastq2: One of the constraints is the need for a RedHat or CentOS installation to use the bcl2fastq2 tool; building from the source was no longer working. Illumina provides a detailed manual so that molecular biology researchers can use it without specialized training. But of course, the original BCL files are needed to use it, which often are not given away by the sequencing facilities.
- Illumina2bam: Illumina2bam has a one-page website as documentation which is sufficient for most operations. Details about the demultiplexing itself are not disclosed and can only be inferred by the behavior of the tool when switching the error levels; therefore, reproducibility may be jeopardized. Setup requirement is java8, and it has to be installed on the client, newer versions tend to have problems with some functions of the tool.
- Python Levenshtein/mutations: Only python 3.7 is needed as setup; all

requirements are described and ready to install in a requirements file. The command-line tool has an inbuild help function and has sanity checks in case some parameters are missing or, for example, an extensive amount of errors is found in the provided files. Detailed metrics are printed out at the end of each run. The interviewed molecular biology Ph.D. tested the application, and he was able to use it without any prior training or help. After the first tests he suggested some metrics like basic error rates of the barcodes as well as the overall number of matched sequences per barcode to be included in the output.

- SQLite/PostgreSQL/MongoDB: This application is similar to the Python Levenshtein/mutations prototype. The only difference is that it consists of two modes, which are automatically switched by the program itself.

    1. If no database is present, then the program will start building up the database consisting of the sequences and barcodes provided by the user.
    2. If a database with the desired sequences is detected, then the program starts the demultiplexing process.

    The database to be used can be adjusted in the settings of the program or is an optional parameter, where SQLite is the default value. In case PostgreSQL or MongoDB are selected, a database server has to be provided. This is a downside to this approach since the questionnaire and the interview indicated that databases would need to be provided by the IT department and cannot be installed by the scientists on their own.

If available, Docker, singularity or podman can be used to run the python prototypes as well as the databases itself, Dockerfiles are provided with instructions. The interviewee from the qualitative interview was able to run the Docker-based versions without prior training or help.

8 HiC Pipeline

After generating the sequences from the HiC method, the raw data has to be analyzed to get the requested insight into the genome structure. Figure 13 provides an overview of the steps involved. The detailed steps are as followed:

1. **Demultiplexing with bcl2fastq2 [18]**: If the Sequencing facility does not provide you with demultiplexed.fastq or unalligned .sam/.bam files this step is necessary to distinguish between the individual samples (often different experimental conditions) and pooled samples.
2. **FastQC (on each sample) for raw sequencing reads quality control**: FastQC is a tool that provides series of quality control checks on raw sequence data. It provides a report including visualizations, to give a quick impression if the data has any heavier problems one should be aware of or consider,before doing any further analysis [42].

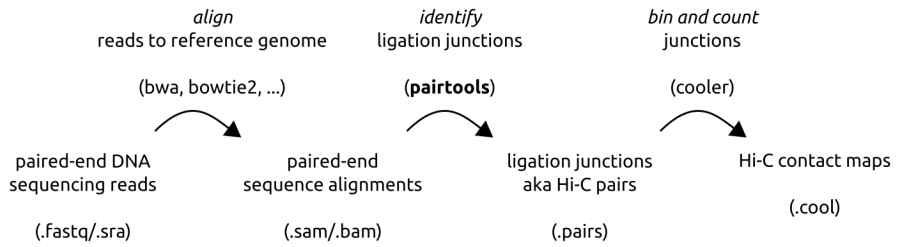

Figure 13: HiC Pipeline [4]

3. **Merge the fq-files of two lanes**: Since a full Illumina Novaseq run includes two flowcells, a simple operation that concatenates two gziped FASTQ files performed with the unix 'cat' command.

4. **Alignment using bwa mem[29][27]**: The reads were aligned to the hg19 human genome [37] with a correction for aberrant genome situation (heSNPs and chromosomal translocations) of our HeLa cervical cancer cell line used in our experiments. Unfortunately the bwa mem algorithm has not yet been published.

5. **Parse**: find ligation junctions in .sam, make .pairsam

6. **Sort**: Sorting the .pairsam files

7. **Dedup**: Find and remove PCR/optical duplicates. We used pairtools [4] to detect ligation junctions of HiC pairs in aligned paired-end sequences of Hi-C DNA molecules (done by parse), These .pairs files were sorted for downstream analyses and PCR/optical duplicates were detected, tagged and removed.

8. **Annotate S4T mutations**: We labeled a subset of DNA using synthetic nucleotides, in our case the newly synthesized sister chromatid and developed a novel method coined sister-chromatid sensitive Hi-C (scsHi-C) that allows the elucidation of sister chromatid structure at unprecedented resolution. During this step we were looking for the characteristic T-to-C and A-to-G mutations and decided afterwards, according to heuristics, if the mutations are due to the labeling.

9. **Filter cis and trans contacts**: Filter for contacts within or between sister chromatids.

10. **Merge ref and comp .pairs file**: Merge reference and complementary strand reads using pairtools [4].

11. **Generate cools**: Create a cooler (i.e. .cool files) from genomic pairs and bins, which is an efficient storage format for high resolution genomic interaction matrices using cooler [1].

12. **Zoomify and balance**: generates a multi-resolution cooler file by coarsening and out-of-core matrix balancing. It is the final output and is loaded by the HiGlass visualization system [22]

8.1 Nextflow HiC Pipeline

J. Paul Rodker Morrison first described the concept of flow based programming in the late '60s [34]. The core concept behind it, is that an application is just a network of black-boxed processes that communicate with one another via predefined connections. Since these processes are isolated in themselves, this concept is component-oriented. Nextflow is an excellent example of a framework that works within the idea of flow based programming.
Nextflow is a pipelining tool specialized in genomic pipelines; it was published by Di Tommaso Paolo et al. [10], but existed as an open-source project a while before that. As the survey shows, it is well known in the community and is used to combine multiple smaller tools into an efficient pipeline to automate most of the work during the analysis process. Each step in the pipeline is set in a Nextflow file that defines a pipelines logic written in groovy. Furthermore, there are configuration files that allow us to define an executor, this provides an abstraction to the execution layer so that it can be executed on multiple platforms without modifications. In our case this opens the opportunity to run the pipeline locally on a batch system or on AWS without any modification of the pipeline. The tool support, mostly cloud/grid or batch setups and their scheduler systems, is used in scientific computing settings.
Another feature is the support of containers, allowing to run any pipeline component in Docker or Singularity containers, and any software on rootless environments without help and support from IT departments.
Furthermore, all the data chunks that the system produces are tracked, which allows to resume its execution, from the last successfully executed step during development or on unstable cluster conditions.
The tool itself supports reports that summarize resource usage and execution steps for the individual components to early identify bottlenecks during the development.

8.2 MapReduce

MapReduce is a programming paradigm that was developed by Google, Inc. and published in 2004 by Jeffrey Dean and Sanjay Ghemawat [9]. It describes a specific data workflow which is split up into three phases:

1. Map: The data gets distributed among the Map processes and a user-defined function, afterwards it maps each item to a key.

2. Shuffle: These key-value pairs then get sorted according to their key. Each key is at least one reduce process.

3. Reduce: The reducer executes another user-defined function onto the data. After this, the results either get saved or given to another Map step.

Each of the Map and Reduce processes can be executed on its own computing node. Because of this, every problem that can be expressed within this paradigm, can be executed in a parallel mode.

8.3 Apache Spark

Apache Spark was developed and published by Zaharia et al. at the University of California, Berkeley in 2010 [50]. It is viewed by most people as the successor to Hadoop as a MapReduce framework. It has new features that ensure fault tolerance as well as overall better runtime.
The two main principles of Apache Spark besides the MapReduce paradigm are:

- Keeping data in memory to ensure faster access and only send the final results back to the disk.

- The second principle of lazy evaluation directly enhances this. Modifications to the data are not done immediately; they are collected and only executed if necessary by the workflow, ensuring optimization of the accumulated changes.

The data is stored in Resilient Distributed Datasets (RDDs), which are read-only. Each worker has its own set of data, to get a final result in the driver program, the intermediate results of the workers have to be collected. Currently, Apache Spark API supports Scala, Java, and Python as higher-level languages.

8.4 Performance

Regarding to performance matters, using Apache Spark-based Frameworks in the HiC Pipeline has the potential to provided an immense speedup. In this case I was able to run the the most time consuming part of the pipeline, the alignment with bwa mem in parallel. For accurate timings future testing on a stable setup is required. In addition, tuning and testing are required to achieve the best possible performance. That promises that it could be a feasible strategy to use state of the art technologies like Apache Spark in the field of molecular biology.

8.5 Usability

In order to run Apache Spark efficiently, a Spark cluster with worker nodes has to be set up before submitting the job to the master node.
Unfortunately, all tested Spark frameworks were challenging to compile or run.

Documentation or support was difficult to find, details on the different frameworks are described in the chapter challenges and lessons learned. After about 20 workhours the Adam framework [32] finally worked as intended.
The amount of work and knowledge that is necessary in order to run Spark frameworks for next generation sequencing problems makes them as standalone tools simply neither practical nor feasible. Some wrapper is needed in order to make these tools more accessible for the researcher that would benefit from the performance.

9 Challenges and Lessons Learned

During the development of the original demultiplexing tool, I already encountered several challenges. I learned a lot about how scientists work within the molecular biology area and how best practices are applied outside of computer science topics:

- The documentation of most tools I stumbled across, seemed to lack necessary usage information and were often merely a patchwork text of different commands rather than information on what the tool's functions are, not to talk about an explanation of every parameter and flag. An excellent example for this matter is StreamAligner. There, the documentation only exists of multiple lines of starting commands [40]. This has also an impact on the reproducibility, since tools like the Burrows-Wheeler Aligner are indeed commonly used, but their algorithm is not published; the consequence is, that the analysis method is not precisely definable with formulas but rather a chain of tools.

- Maintenance is also a big issue in the community of bioinformatics, since a lot of tools are no longer maintained. It is often hard to get them running without having to create a virtual environment through already downgraded software. SparkBWA, for example, wouldn't compile anymore unless Java version 8 is used and an older version of Hadoop and Spark [2].

- Additionally, the required setup was not always fully disclosed, and most tools had no section on how to install or compile. An extensive review was already done on this issue by Mangul et al. in 2019 [31].

- Communication and Terminology is also a challenge I encountered while developing tools in the field of bioinformatics. In the survey, more than half of the researchers said they use databases, but the interview later-on revealed that they probably meant genome databases rather than databases in a computer science sense.

10 Future Work

The need for qualitative analysis software was revealed during the time working on different problems at the Vienna Bio Center. Most researchers have neither the knowledge nor the time to go through an extensive review process before choosing a fitting tool. The cost being that they are often unable to determine exactly what a tool is able to do regarding the ensurance of reproducibility. Another result is that state of the art technologies and solutions are not used which hinders the development of better tools.
A possible solution and the next step to improve data analysis in molecular biology research would be to develop a useful tool which helps scientists to find the adequate software for their specific problem and which supports them during the process of building their individual pipeline. It should provide information on how their output is calculated, on their overall performance, and compatibility with other tools considering the different possible input and output file formats shown in chapter 5.
It should help the researcher build the pipeline as a form of questionnaire, where the researcher then can exchange different steps in the pipeline in kind of a plug and play manor. Helpful would be, if the tools would be provided as containers so that portability can always be guaranteed. Ideally, it is not limited to one pipelining tool; instead, it should support a variety of them, starting with Nextflow as it is one of the most used pipelining tools in the next generation sequencing sector.
Without such specialized tools research in this field will not have the possibilities to continue on a professional level. Results can no longer be reproduced from the raw data and credibility is only a matter of trust rather than of skill [14].

11 Appendix

To get a better understanding on how and why so much data is created by next-generation sequencing methods and sequencing in general the basic molecular biology concepts are explained in the following sections.

11.1 PCR

These are the components required for every PCR:

- the template DNA: this is the whole DNA sequence with the target sequence in it
- primers: short DNA sequences that can be bound to the start and end of the target sequence and start the PCR
- DNA base pair: sufficient base pair (A, C, G, T) to build the new copies of the target sequence

- buffer solution: the buffer solution is needed to ensure an environment that resembles the cell, therefore has a pH between seven and ten, otherwise the bounds between the DNA pairs wouldn't hold and the catalysis wouldn't run. Usually NaCl, MgCl, Tris-Cl.
- Taq polymerase enzyme: these thermostable polymerase enzymes are required to bind the DNA base pairs together, it does this by matching the right bases according to the template DNA.

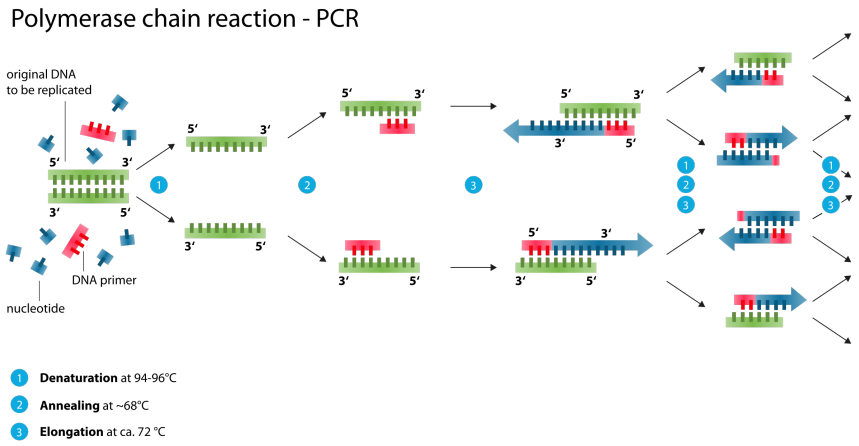

Figure 14: PCR-Cycle [11]

PCR has the following steps[35][48] as seen in figure 14:

1. Denaturation: the double-stranded template DNA is heated to about 94-96°C to denature, that means the hydrogen bonds between the strands break and so it separates into two single strands.
2. Annealing: the primers can now attache to the single strands, which is initiated by cooling the solution to about 68°C. One primer, the forward primer, attaches itself to the beginning of the target strand. The other primer, the reverse primer, attaches itself to the end of the strand.
3. Elongation: the Taq polymerase enzyme now can use the primers as an anchor point and will start to replicate the separated strands by building a new one from the base pairs in the solution, complementary to corresponding strands, following the base pairing rule.
    - A<->T
    - G<->C

These steps are repeated several times, by which the target DNA is doubled every iteration step.

11.2 Sanger Sequencing

Sanger sequencing consists of the following steps as shown in figure 15:

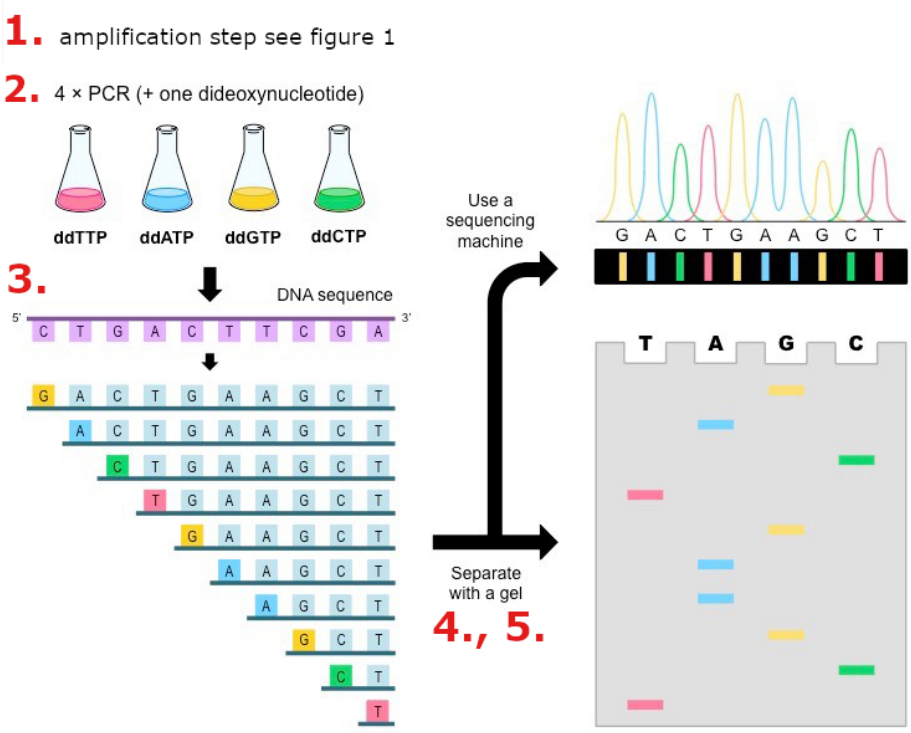

Figure 15: Sanger-Sequencing adapted from [21]

1. Amplification: Amplifying the target DNA: In the original paper from 1977 by Sanger [41], a linear amplification by cloning via bacteria is used. After 1986 regular PCR as described above was used instead of the linear amplification.

2. Split: Splitting up the amplified strands into four groups and adding respectively the four different base pairs.

3. Dideoxynucleotide bases: Adding dideoxynucleotide bases to a radioactive or fluorescent tag which terminate the polymerase at a certain point. Only one respective dideoxynucleotide bases is added to one of the four groups that ensures that only one specific base has the chance to terminate. Today

this is not necessary anymore, the DNA usually gets stained inside the gel in the next step.

4. Electrophoresis: Electrophoresis splits up the different subsequences, which are now unequal in length, depending on where the dideoxynucleotide bases are attached. That works by attracting the negative sequences to the positive pole and drag them through argarose-gel.

5. Evaluation: In the end, four different lanes with detectable markers should be observable. Looking at these lanes from bottom to top results in a reverse read of the target DNA.

11.3 Illumina Method

These are the steps of the Illumina Method:

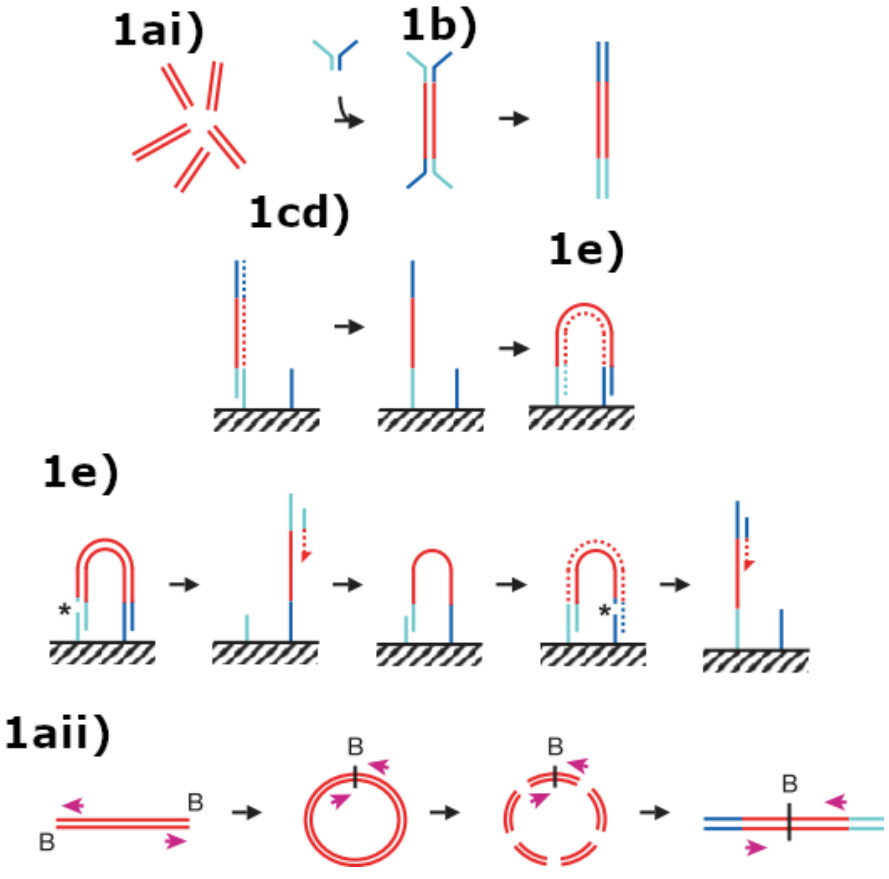

Figure 16: Illumina Sample Preperation adapted from [5]

1. Preparation steps as seen in figure 16:

   (a) i. Shotgunning: First, the DNA gets "shotgunned", which means that it gets split up into smaller pieces, for example by random shearing, that are about the size of 200-600 base pairs.

   ii. Circulatization of long sequences: An alternative to shotgunning for longer sequences (more than one thousand base pairs) to be able to read their ends, is to circularize them by combining beginning and end. After that, it also gets cut down into smaller pieces, but only the pieces that where labelled by the biotin get pulled down and can be used for the next steps, all other pieces are washed away. This special handling of longer sequences allows us to read them more efficiently since begin and end are read in parallel and also increases the read accuracy

   (b) Oligonucleotides: Adapters, two different complementarities to the flowcell oligonucleotides are added to either side of the sequence. In this step, additional information can be added, before the complementarities, often the barcodes are added as indices to the specific sequences. Commonly all sequences from an experiment are labeled with the same one to two barcodes.

   (c) Annealing: These sequences are then annealed to the corresponding oligonucleotides on the flowcell.

   (d) Forming a complementary strand: Now with polymerase, a new strand is formed starting on the oligonucleotides on the flowcell. After that, the original strand is denatured, and so only a complementary strand is left on the flowcell.

   (e) Bridge amplification: Since on strand alone would be too small to see, the single strand now folds over and is annealed with the complementary oligonucleotides from the other end of the sequence.

   (f) Polymerization of the bridge: This newly formed bridge between the two oligonucleotides can then be cloned again via polymerase.

   (g) Formation of a clonal single-molecule array: Step e and f are then repeated a few million times until the now forming clusters or socalled clonal single-molecule array, with a physical diameter of about 1 μm form. After this step, a known problem is the mixing of two different sequence pieces, when two clusters start to grow in one another. This issue has then to be resolved in the next steps of sequencing.

2. Sequencing steps:

   (a) Washing away the reverse reads: First, all reverse reads get washed away, so only reverse reads are left on the flowcell.

   (b) Adding bases: In this step, all bases are added to the flowcell modified so that they can be excited by a light source and each base then

emits a specific light wavelength. Only one base pair at a time can be attached to the forward read because the modified bases block the attachment of new bases.

(c) Exciting and light analysis: This newly formed sequence is then excited by a laser and emits, as seen in figure 17, one of four or three lights, depending on how many channels the machine uses for exciting. A known problem hereby, despite being advertised as a superior method [19], is that read errors and the base G are not distinguishable anymore, leading to an overall higher error rate and making analysis on this data harder since information is no less reliable. Also, the previously mentioned problem of two clusters merging is now not as easy to detect as if all four channels are used.

(d) Washing away the modification: As of the last step, the light-emitting modification of the base is washed away. With it gone the base no longer blocks the attachment of a new base.

(e) Step 2b-2d are then repeated n times, where n is the number of bases in the sequence.

(f) Reading the index: Now the index has to be read in the same fashion as the whole forward read was. The index is usually six base pairs long.

(g) For the reverse read, bridge amplification is used one more time as described in step 1e and 1f. After that, the forward read is washed away, and the reverse read is then sequenced like the forward read described in 2b-2d.

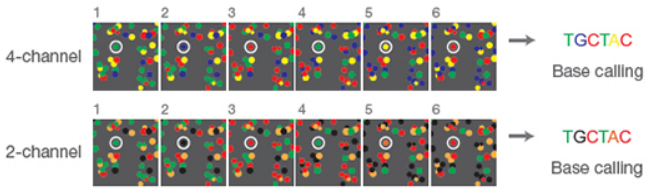

Figure 17: Illumina 2 and 4 channels adapted from [19]

11.4 Online Questionnaire

The online questionnaire can also be found under:
https://forms.gle/6Bu1dYNFu219tKQ47 .

11.4.1 General Questions

In this sections the participants where asks to answer some basic questions about their position and daily work.

What's your position in your research group/department? Single Choice

- Master Student
- PhD
- Research Assistant
- PostDoc
- Research Associate
- Staff Scientist
- Group Leader
- Other(writable)

From which journals do you get information about bioinformatics tools? Open Question

How do you choose your bioinformatics tools? Multiple Choice

- Modifying existing legacy pipelines in the lab
- Recommendations from colleagues and collaborators
- Googling
- Novel approaches from bioinformatics literature
- Other(writable)

What describes you best? Single Choice

- Tool/pipeline user
- Pipeline creator
- Tool modifier
- Tool creator
- Other(writable)

What applications do you use bioinformatics analysis for? Multiple Choice

- RNAseq
- Chip-seq
- Genomics
- Predictive models
- Image analysis
- Other(writable)

How much is your time split between bioinformatics and wet-lab work Likert Scale [1: All bench,...,10: All computational]

How would you rate your programming skills? Single Choice If "No programming skills" was selected section 2 was skipped.

- No programming skills
- Beginner
- Intermediate
- Expert

11.4.2 Programming Questions

In the second section the focus was to determined the programming background and knowledge of the participants

What's your to go programming language/s? Multiple Choice

- Python
- R
- C++
- C
- Java
- Matlab
- Other(writable)

Which of them do you use most? Open Question

How did you learn programming? Multiple Choice

- Self taught
- From colleagues or lab members
- Online course
- School
- University courses
- Workshops
- Computer Science degree
- Other(writable)

Do you use any of the following concepts? Multiple Choice

- Databases
- Map and Reduce
- Concurrency / Parallel Computing
- Machine Learning
- Microservices
- Distributed Systems
- Containers
- Virtual Environments
- Other(writable)

If yes, in which way and use case? Open Question

Do you use any of the following framworks/systems? Multiple Choice

- Conda
- Singularity/Docker/Podman
- Nextflow
- Snakemake
- Apache Spark/Hadoop
- Cloud Services (Google Cloud, Amazon Cloud, Microsoft Azure)
- Other(writable)

If yes, in which way and use case? Open Question

Which software practices do you regularly (weekly) use? Multiple Choice

- regular backups
- source code control
- test driven development
- continuous integration server
- collaborative wikis
- issue tracking
- code reviews
- Other(writable)

The software I develop is for Multiple Choice

- Me only
- Inter lab usage
- Inter institute usage
- opensource repository
- explicit bioinformatics publication
- Other(writable)

11.4.3 Questions about future tools and usability

In the last section questions about software tool features and usability were asked.

Which interface or interaction is required to use your tools? Multiple Likert Scales [Not important; Nice to have; Important; Must]

- Graphical User Interface
- Command Line Interface
- Only library/has to be integrated

In which form do you want your results? Multiple Likert Scales [Not important; Nice to have; Important; Must]

- Output to the terminal
- One single result file
- All intermediate files
- Visualization

How do you choose between multiple tool options? Multiple Likert Scales [Not important; Nice to have; Important; Must]

- Large community
- Known programming language
- Detailed documentation
- Tutorial
- Video demo
- Last update
- Open source
- Fast runtime
- Good compatibility with existing tools/pipelines
- Fast setup time
- Minimal User Interaction (Full Automation)
- Ability to adapt steps & configs

How much of your time are you waiting for jobs to finish? Open Question

If you had time to implement certain automation of your daily tasks, how much of your time would be freed for other projects? Open Question

What is the greatest bottleneck in your analysis workflow? Open Question

What is the one thing that you think could advance the field of bioinformatics in the next year? Open Question

References

[1] Abdennur, N., and Mirny, L. A. Cooler: scalable storage for Hi-C data and other genomically labeled arrays. *Bioinformatics* (07 2019).

[2] Abuín, J. M., Pichel, J. C., Pena, T. F., and Amigo, J. SparkBWA: Speeding up the alignment of high-Throughput DNA sequencing data. *PLoS ONE 11*, 5 (2016).

[3] Andreas Heger, Kevin Jacobs, *et al.*. Pysam Python Library. `https://pysam.readthedocs.io/en/latest/api.html`. [Online; accessed August 08, 2019].

[4] Anton Goloborodko. Pairtools. `https://github.com/mirnylab/pairtools`. [Online; accessed August 18, 2019].

[5] Bentley, D. R., Balasubramanian, S., Swerdlow, H. P., Smith, G. P., Milton, J., Brown, C. G., Hall, K. P., Evers, D. J., Barnes, C. L., Bignell, H. R., Boutell, J. M., Bryant, J., Carter, R. J., Keira Cheetham, R., Cox, A. J., Ellis, D. J., Flatbush, M. R., Gormley, N. A., Humphray, S. J., Irving, L. J., Karbelashvili, M. S., Kirk, S. M., Li, H., Liu, X., Maisinger, K. S., Murray, L. J., Obradovic, B., Ost, T., Parkinson, M. L., Pratt, M. R., Rasolonjatovo, I. M., Reed, M. T., Rigatti, R., Rodighiero, C., Ross, M. T., Sabot, A., Sankar, S. V., Scally, A., Schroth, G. P., Smith, M. E., Smith, V. P., Spiridou, A., Torrance, P. E., Tzonev, S. S., Vermaas, E. H., Walter, K., Wu, X., Zhang, L., Alam, M. D., Anastasi, C., Aniebo, I. C., Bailey, D. M., Bancarz, I. R., Banerjee, S., Barbour, S. G., Baybayan, P. A., Benoit, V. A., Benson, K. F., Bevis, C., Black, P. J., Boodhun, A., Brennan, J. S., Bridgham, J. A., Brown, R. C., Brown, A. A., Buermann, D. H., Bundu, A. A., Burrows, J. C., Carter, N. P., Castillo, N., Catenazzi, M. C. E., Chang, S., Neil Cooley, R., Crake, N. R., Dada, O. O., Diakoumakos, K. D., Dominguez-Fernandez, B., Earnshaw, D. J., Egbujor, U. C., Elmore, D. W., Etchin, S. S., Ewan, M. R., Fedurco, M., Fraser, L. J., Fuentes Fajardo, K. V., Scott Furey, W., George, D., Gietzen, K. J., Goddard, C. P., Golda, G. S., Granieri, P. A., Green, D. E., Gustafson, D. L., Hansen, N. F., Harnish, K., Haudenschild, C. D., Heyer, N. I., Hims, M. M., Ho, J. T., Horgan, A. M., Hoschler, K., Hurwitz, S., Ivanov, D. V., Johnson, M. Q., James, T., Huw Jones, T. A., Kang, G. D., Kerelska, T. H., Kersey, A. D., Khrebtukova, I., Kindwall, A. P., Kingsbury, Z., Kokko-Gonzales, P. I., Kumar, A., Laurent, M. A., Lawley, C. T., Lee, S. E., Lee, X., Liao, A. K., Loch, J. A., Lok, M., Luo, S., Mammen, R. M., Martin, J. W., McCauley, P. G., McNitt, P., Mehta, P., Moon, K. W., Mullens, J. W., Newington, T., Ning, Z., Ling Ng, B., Novo,

S. M., O'Neill, M. J., Osborne, M. A., Osnowski, A., Ostadan, O., Paraschos, L. L., Pickering, L., Pike, A. C., Pike, A. C., Chris Pinkard, D., Pliskin, D. P., Podhasky, J., Quijano, V. J., Raczy, C., Rae, V. H., Rawlings, S. R., Chiva Rodriguez, A., Roe, P. M., Rogers, J., Rogert Bacigalupo, M. C., Romanov, N., Romieu, A., Roth, R. K., Rourke, N. J., Ruediger, S. T., Rusman, E., Sanches-Kuiper, R. M., Schenker, M. R., Seoane, J. M., Shaw, R. J., Shiver, M. K., Short, S. W., Sizto, N. L., Sluis, J. P., Smith, M. A., Ernest Sohna Sohna, J., Spence, E. J., Stevens, K., Sutton, N., Szajkowski, L., Tregidgo, C. L., Turcatti, G., Vandevondele, S., Verhovsky, Y., Virk, S. M., Wakelin, S., Walcott, G. C., Wang, J., Worsley, G. J., Yan, J., Yau, L., Zuerlein, M., Rogers, J., Mullikin, J. C., Hurles, M. E., McCooke, N. J., West, J. S., Oaks, F. L., Lundberg, P. L., Klenerman, D., Durbin, R., and Smith, A. J. Accurate whole human genome sequencing using reversible terminator chemistry. *Nature 456*, 7218 (2008), 53–59.

[6] Brewer, E. A. Towards robust distributed systems. In *Proceedings of the nineteenth annual ACM Symposium on Principles of Distributed Computing - PODC '00* (New York, New York, USA, 2000), ACM Press, p. 7.

[7] Broad Institute of MIT and Harvard. Picard Tools. `https://broadinstitute.github.io/picard/`. [Online; accessed August 14, 2019].

[8] Cock, P. J. A., Fields, C. J., Goto, N., Heuer, M. L., and Rice, P. M. The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. *Nucleic Acids Research 38*, 6 (2009), 1767–1771.

[9] Dean, J., and Ghemawat, S. Simplified data processing on large clusters. *6th Symposium on Opearting Systems Design & Implementation 51*, 1 (2004), 107–113.

[10] Di Tommaso, P., Chatzou, M., Floden, E. W., Barja, P. P., Palumbo, E., and Notredame, C. Nextflow enables reproducible computational workflows. *Nature Biotechnology 35*, 4 (4 2017), 316–319.

[11] Enzoklop, and Wikimedia Commons. Polymerase chain reaction. `https://commons.wikimedia.org/wiki/File:Polymerase_chain_reaction.svg`, 2014. [Online; accessed July 22, 2019].

[12] Fjukstad, B., and Bongo, L. A. A Review of Scalable Bioinformatics Pipelines. *Data Science and Engineering 2*, 3 (2017), 245–251.

[13] Forward, A., and Lethbridge, T. C. The Relevance of Software Documentation, Tools and Technologies: A Survey. Tech. rep., 2002.

[14] Grüning, B., Chilton, J., Köster, J., Dale, R., Soranzo, N., van den Beek, M., Goecks, J., Backofen, R., Nekrutenko, A.,

and Taylor, J. Practical Computational Reproducibility in the Life Sciences. *Cell Systems 6*, 6 (2018), 631–635.

[15] Heng Li, and Wellcome Sanger Institute. FASTQ Format Specification. `http://maq.sourceforge.net/fastq.shtml`. [Online; accessed July 25, 2019].

[16] Hipp, Wyrick & Company, Inc., (Hwaci). SQLite Documentation. `https://www.sqlite.org/docs.html`. [Online; accessed August 14, 2019].

[17] Ian Dickson. Database Guide. `https://database.guide/`. [Online; accessed August 14, 2019].

[18] Illumina, Inc. bcl2fastq and bcl2fastq2 Conversion Software. `https://support.illumina.com/sequencing/sequencing_software/bcl2fastq-conversion-software.html`. [Online; accessed August 14, 2019].

[19] Illumina, Inc. Faster sequencing and data processing 2-channel SBS generates data faster than 4-channel SBS, while maintaining quality and accuracy. `https://www.illumina.com/science/technology/next-generation-sequencing/sequencing-technology/2-channel-sbs.html`, 2015. [Online; accessed July 24, 2019].

[20] Jeff Gilchrist. pbzip2(1) - Linux man page. `https://linux.die.net/man/1/pbzip2`. [Online; accessed August 13, 2019].

[21] Karki, G. Sanger's method of gene sequencing. `https://www.onlinebiologynotes.com/wp-content/uploads/2017/07/dna-sequencing_med.jpeg`, 2017. [Online; accessed July 22, 2019].

[22] Kerpedjiev, P., Abdennur, N., Lekschas, F., McCallum, C., Dinkla, K., Strobelt, H., Luber, J. M., Ouellette, S. B., Azhir, A., Kumar, N., Hwang, J., Lee, S., Alver, B. H., Pfister, H., Mirny, L. A., Park, P. J., and Gehlenborg, N. HiGlass: Web-based visual exploration and analysis of genome interaction maps. *Genome Biology 19*, 1 (2018).

[23] Kevin Sidak. Quantitative Questionnaire. `https://docs.google.com/forms/d/1-G-w0j95d1uXnDz3ydcCRY461cwXmflr7kZm6u8Jy3M/edit`. [Online; accessed August 18, 2019].

[24] Kosicki, M., Tomberg, K., and Bradley, A. Repair of double-strand breaks induced by CRISPR–Cas9 leads to large deletions and complex rearrangements. *Nature Biotechnology 36*, 8 (9 2018).

[25] Lajoie, B. R., Dekker, J., and Kaplan, N. The Hitchhiker's guide to Hi-C analysis: Practical guidelines. *Methods 72*, C (2015), 65–75.

[26] Levenshtein, V. Binary codes capable of correcting deletions, insertions, and reversals. *Doklady Akademii Nauk 163*, 4 (1965), 845–848.

[27] Li, H. Exploring single-sample SNP and INDEL calling with whole-genome De Novo assembly. *Bioinformatics 28*, 14 (7 2012), 1838–1844.

[28] Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., Marth, G., Abecasis, G., and Durbin, R. The Sequence Alignment/Map format and SAMtools. *Bioinformatics 25*, 16 (2009), 2078–2079.

[29] Li H. and Durbin R. Burrows-Wheeler Alignment Tool. `http://bio-bwa.sourceforge.net/bwa.shtml`. [Online; accessed August 18, 2019].

[30] Lieberman-Aiden, E., Van Berkum, N. L., Williams, L., Imakaev, M., Ragoczy, T., Telling, A., Amit, I., Lajoie, B. R., Sabo, P. J., Dorschner, M. O., Sandstrom, R., Bernstein, B., Bender, M. A., Groudine, M., Gnirke, A., Stamatoyannopoulos, J., Mirny, L. A., Lander, E. S., and Dekker, J. Comprehensive mapping of long-range interactions reveals folding principles of the human genome. *Science 326*, 5950 (10 2009), 289–293.

[31] Mangul, S., Mosqueiro, T., Abdill, R. J., Duong, D., Mitchell, K., Sarwal, V., Hill, B., Brito, J., Littman, R. J., Statz, B., Lam, A. K.-M., Dayama, G., Grieneisen, L., Martin, L. S., Flint, J., Eskin, E., and Blekhman, R. Challenges and recommendations to improve the installability and archival stability of omics computational tools. *PLOS Biology 17*, 6 (2019), e3000333.

[32] Massie, M., Nothaft, F., Hartl, C., Kozanitis, C., Joseph, A. D., Patterson, D. A., and Sciences, C. ADAM: Genomics Formats and Processing Patterns for Cloud Scale Computing. Tech. rep., 2013.

[33] MongoDB, Inc. MongoDB Homepage. `https://www.mongodb.com`. [Online; accessed August 14, 2019].

[34] Morrison, J. Flow-Based Programming: A new approach to application development. *Proc. 1st International Workshop on Software ...* (2011), 316.

[35] Mullis, K., Faloona, F., Scharf, S., Saiki, R., Horn, G., and Erlich, H. Specific enzymatic amplification of DNA in vitro: the polymerase chain reaction. *Cold Spring Harbor symposia on quantitative biology 24*, Table 1 (1986), 17–27.

[36] Munzner, T. *Visualization Analysis and Design.* A K Peters/CRC Press, 2014.

[37] National Center for Biotechnology Information. GRCh37 - hg19 - Genome - Assembly - NCBI. https://www.ncbi.nlm.nih.gov/assembly/GCF_000001405.13/, 2013. [Online; accessed June 05, 2020].

[38] National Humane Genome Research Institute. The Cost of Sequencing a Human Genome. https://www.genome.gov/about-genomics/fact-sheets/Sequencing-Human-Genome-cost. [Online; accessed August 14, 2019].

[39] Ramalho, L. *Fluent python: clear, concise, and effective programming.* O'Reilly, 2015.

[40] Rathee, S., and Kashyap, A. StreamAligner: a streaming based sequence aligner on Apache Spark. *Journal of Big Data 5*, 1 (12 2018), 8.

[41] Sanger, F., Nicklen, S., and Coulson, A. R. DNA sequencing with chain-terminating inhibitors. *Proceedings of the National Academy of Sciences of the United States of America 74*, 12 (1977), 5463–7.

[42] Simon Andrews. FastQC. https://www.bioinformatics.babraham.ac.uk/projects/fastqc/. [Online; accessed August 18, 2019].

[43] Stonebraker, M., Madden, S., Abadi, D., Dewitt, D. J., Paulson, E., Pavlo, A., and Rasin, A. MapReduce and parallel DBMSs: Friends or foes? *Communications of the ACM 53*, 1 (2010), 64–71.

[44] Stonebraker, M., Rowe, L. A., and Hirohama, M. The Implementation of POSTGRES. *IEEE Transactions on Knowledge and Data Engineering 2*, 1 (1990), 125–142.

[45] The PostgreSQL Global Development Group. PostgreSQL Documentation. https://www.postgresql.org/docs/. [Online; accessed August 14, 2019].

[46] The SAM/BAM Format Specification Working Group, Bonfield, J., Marshall, J., and Farjoun, Y. SAM (Sequence Alignment / Map) Format Specification. Tech. rep., 2018.

[47] The SAM/BAM Format Specification Working Group, Bonfield, J., Marshall, J., and Farjoun, Y. Sequence Alignment/Map Optional Fields Specification. Tech. Rep. May, 2019.

[48] Wellcome Genome Campus. What is PCR (polymerase chain reaction)? https://www.yourgenome.org/facts/what-is-pcr-polymerase-chain-reaction, 2016. [Online; accessed July 22, 2019].

[49] Wellcome Trust Sanger Institute. Illumina2bam. https://gq1.github.io/illumina2bam/. [Online; accessed August 14, 2019].

[50] Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., and Stoica, I. Spark : Cluster Computing with Working Sets. *HotCloud'10 Proceedings of the 2nd USENIX conference on Hot topics in cloud computing* (2010), 10.