HIGH THROUGHPUT SEQUENCING

Simon Hegele

February 25, 2025

Introduction

A draft of the contents of the lecture and associated practical on high-throughput sequencing by Prof. Marz (Friedrich-Schiller-University of Jena). The contents of the high-throughput sequencing are not static and are often changed from year to year to account for recent developments in research.

High-throughput sequencing has revolutionized genomics and molecular biology research by enabling rapid and cost-effective analysis of vast amounts of DNA or RNA. Here, we present a summary of high-throughput sequencing and the analysis of the resulting data. One key objective of data analysis is sequence assembly, the reconstruction the original genome or transcriptome. Read mapping is another critical aspect, aligning reads to a reference genome or transcriptome for the study of genetic variants and gene expression analysis. Efficient data analysis allows scientists to unravel the complexities of genetic information, providing novel perspectives and advancements in the understanding of fundamental biological processes.

InfoBox 1: Additional resources

- My personal GitHub repository
  There you can find a jupyter-notebook that easily helps you to generate and visualize exemplary graphs as discussed in chapter 5 and the Bowtie algorithm from chapter 6
- The youtube channel of StatQuest
  Nice, easy and sometimes even funny videos on statistics related to bioinformatics
- The original papers
  Don't be scared to read actual scientific literature! The articles on the methods covered here are, for the most part, comparatively easy to understand, especially thanks to their beautiful graphics. Some of them have already been cited tens of thousands of times and can thus be considered some of the classics of bioinformatics.

Use of Artificial Intelligence: Some text passages were generated by ChatGPT. The correctness of their content has been confirmed by additional research.

Contents

Chapter 1

Biological Basics - DNA, RNA and Gene Expression

1.1 Overview on Gene Expression

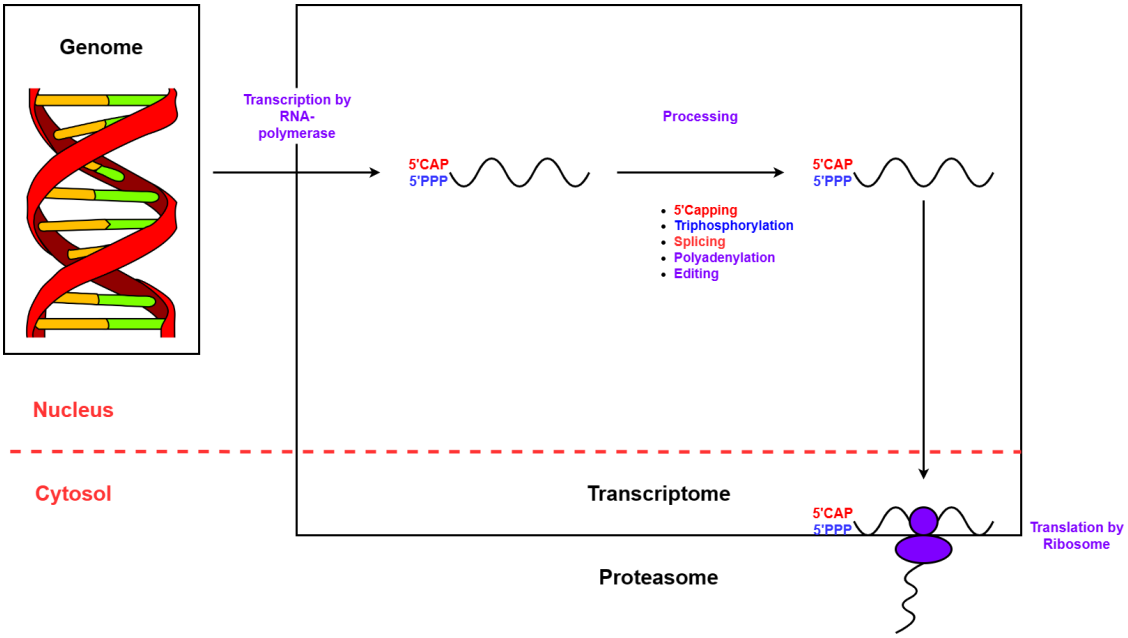

Figure 1.1: Schematic representation of gene expression in Prokaryotes and Eukaryotes
Eukaryotes only
Prokaryotes only
both

Remarks:

- Parallel transcription and processing
- Polyadenylation has different meaning for prokaryotes and eukaryotes
- Are mitochondrial and chloroplastic chromosomes part of the genome?

1.2 Types of DNA

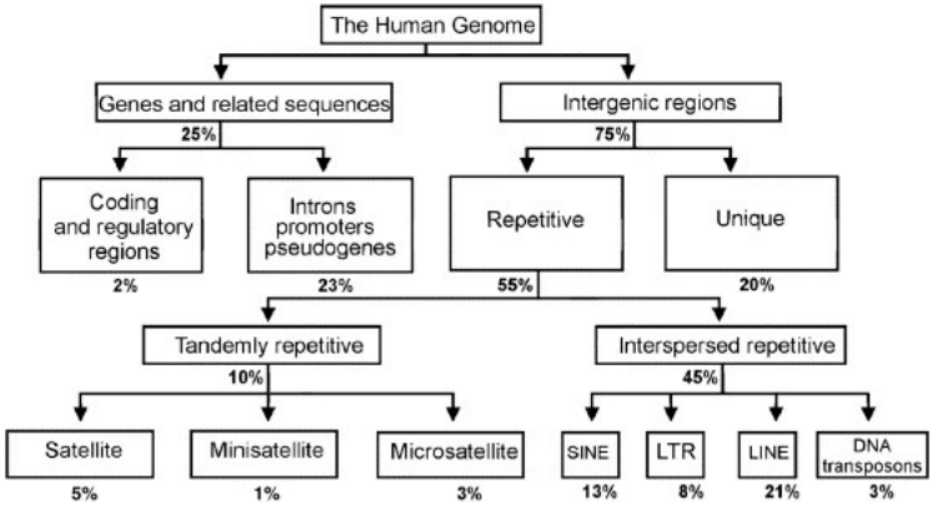

Figure 1.2: Hierarchical organization of sequence types in the human genome with their respective proportions.

1.3 Types of RNA and RNA-Polymerases

1.3.1 RNA

RNA can differ in size, structure and function.

- mRNA (messenger RNA)
- ncRNA (non-coding RNA)
  - rRNA (ribosomal, $\sim 95\%$ of RNA in a cell)
  - tRNA (transfer RNA)
  - snRNA (small nuclear RNA)
  - miRNA (micro RNA)
  - piRNA (piwi-interacting RNA)
  - snoRNA (small nucleolar RNA)
  - ...

1.3.2 RNA-Polymerases

Eukaryotes

Table 1.1: The role of different eukaryotic RNA-polymerases

	Standard knowledge	New knowledge
I	rRNA	
II	mRNA	polyadenylated RNA
III	snc-RNA	

Choice of RNA-polymerase for a gene is determined by its promotor.

Prokaryotes

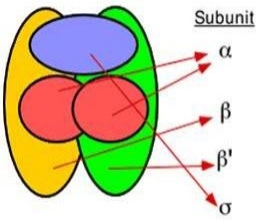

Figure 1.3: Sub-units of prokaryotic RNA-polymerase

The different variants of the σ-factor can be exchanged in the RNA polymerase and serve to recognise different promoters.
σ^x refers to the σ-factor with molecular weight x.

Chapter 2

The Human Genome Project

> "The Human Genome Project has revealed our shared inheritance. It has shown our tiny differences from each other. It has strengthened our understanding of the diversity and unity of humankind."
>
> - A quote that ChatGPT made up and attributed to Mary-Claire King

The Human Genome Project (HGP) publicly funded project initiated in 1990 in order to sequence the human genome, identifying genomic features such as single nucleotide polymorphisms and map them. It was performed by an international consortium of thousands of researchers and was funded for 15 years with $3B.

InfoBox 1: Goals of the the HGP

- Assembly of the genome
- Localization of genes
- Identification of gene functions
- Identification of associations with diseases
- Studying gene interactions
- Studying chromosomal interactions
- Finding gene variants
- Improving sequencing techniques
- Therapy and diagnostics
- Forensic
- Raising an army of bounty hunter clones to overthrow the Republic and destroy the Jedi

InfoBox 2: Ethical questions raised by the HGP

- Data protection
- Discrimination
- Psychological stress due to possible diagnoses
- Risks that come with commercialization

Chapter 3

Sequencing Technologies

Definition 1: Sequencing

RNA- / DNA- Sequencing is the process of determining a nucleotide sequence of a RNA- / DNA- molecule

Definition 2: k-mer

A sequence of length k

Definition 3: Raw read

Sequence(-probability) of a RNA- / DNA- fragment

3.1 RNA-Prep

RNA-preparation is an important step in sequencing experiments and its specific procedure must be adapted to the research questions which can sometimes be addressed by looking only at specific types and conditions of RNA.

Almost universally it is necessary to perform a **ribosome-depletion** because the vast abundance and little informative value of rRNA.

Procedure

1. Lysis of the cell
2. Extraction
    - Using a specialized kits
    - Using phenol-chloroform (For mono-pore sequencing)
3. Filtering (optional)
    - poly-A-Seq
    - ribosome depletion

- small RNA-Seq
- length

(combinations possible)

3.1.1 2nd Generation - Illumina

1. Sample Preparation
   (a) Extraction
   (b) Filtering
   (c) Reverse transcription (for RNA)
   (d) Fragmentation (enzymatic, mechanic,...)
   (e) Adapter ligation
   (f) Amplification (PCR)
2. Cluster formation by bridge amplification
3. Sequencing by synthesis

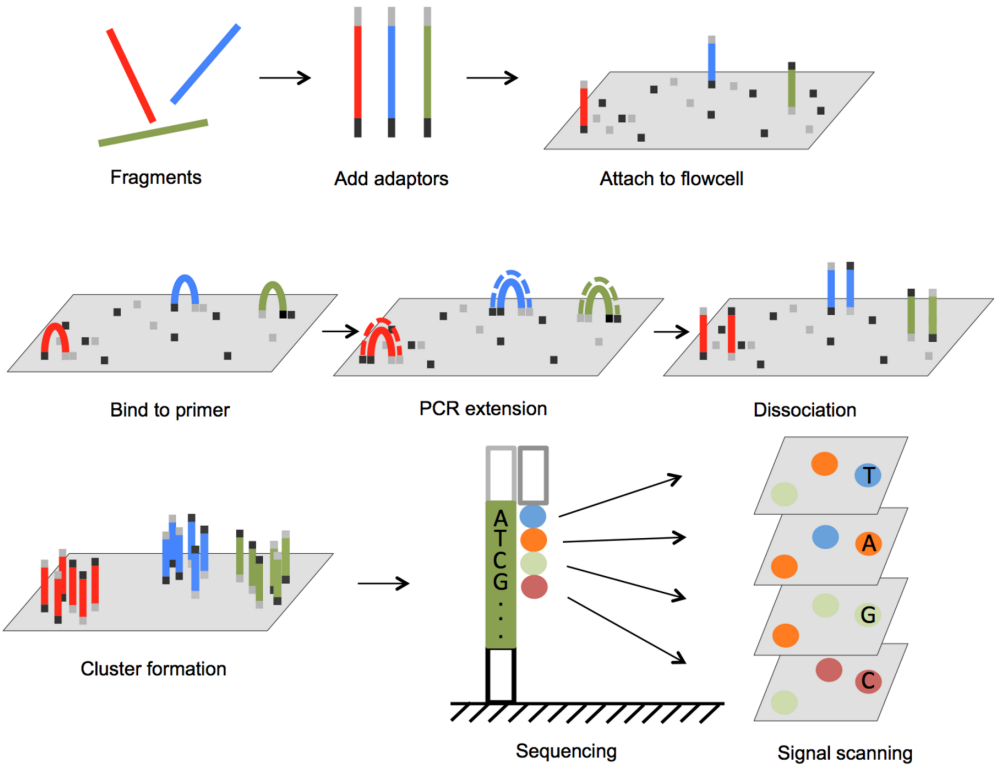

Figure 3.1: Schematic depiction of the procedure of a 2nd generation sequencing experiment

3.2 3rd Generation - ONT and PacBio

Long read sequencing methods produce reads averaging on 10 kb (< 100.000 nt)

ONT (Oxford Nanopore Technologies)

Can detect nucleotide modifications [1].

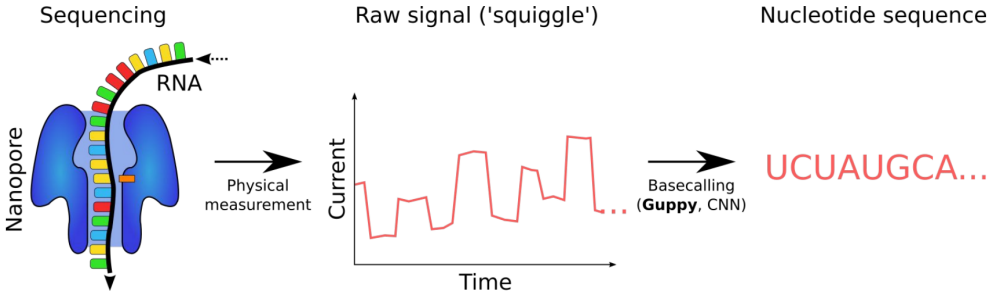

Figure 3.2: Shematic explanation of the ONT-sequencing workflow. An electrical current is applied to a membrane with integrated nanopore. The nanopore allows electrons to pass. Using specialized motor-proteins, single stranded DNA-molecules are also pulled through that nanopore. The electron flux is impacted by the nucleotides that block the nanopore while passing. The current is measured over time. From this raw signal, the nucleotide sequence is derived, since specific sequences leave their specific signature in the raw signal. With the raw signal being very noisy, this is a difficult tasks. Base-calling software often use machine learning methods.

3.2.1 PacBio (Pacific Biosiences)

Individual, single stranded DNA molecules are sequenced by synthesis in small holes by ligation to modified nucleotides which emit light upon binding which is then detected by sensitive photo-receptors.

Watch the very comprehensive video by Pacific Bioscience itself!

3.3 Comparison of 2nd and 3rd Generation sequencing

	2nd Generation	**3rd Generation**
Read length	Short ($\sim 50 - 300$ nt)	Long (~ 100.000 nt)
Read quality	99.9 %-99.99 %	90 %-99.7 % (ONT), 99 %-99.9 % (PacBio)
Throughput	Gigabases	Terabases
Preparation	Multiple enzymatic digestion steps	Simple preparation
Amplification	Yes (Causes Amplification bias and errors)	No
Required coverage	High	Low
Reverse transcription for RNA-Seq	Yes	No
Modifications	Possible but complicated	Possible and easy

Table 3.1: Comparison of sequencing techniques (Some of these values are old and overcome)

Chapter 4

Quality Control and Read Preprocessing

4.1 Quality Control

4.1.1 FastQC

FastQC is a program designed to spot potential problems in high througput sequencing datasets. It runs a set of analyses on one or more raw sequence files in .fastq or .bam format and produces a report which summarises the results.

(Example records)

4.1.2 Read quality

We can measure expected base-calling error probabilities sequencing libraries of known reference sequences

---

Definition 4: Phred score

Let be P be the base-calling error probability of a nucleobase. Then

$$Q = -10 * log_{10}(P)$$

is it's Phred score

Read-Quality in 2nd Generation Sequencing

Quality here depends on the clearness and separability of the light signals from the flow cell. Light signals might mix when clusters aren't spatially separated. The quality drops because the synthesis becomes increasingly asynchronous.

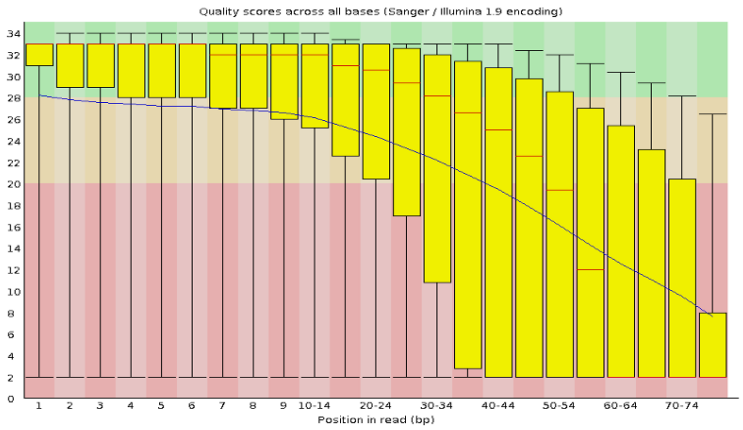

Figure 4.1: Plot from the tool FastQC

Read-Quality in 3rd Generation Sequencing

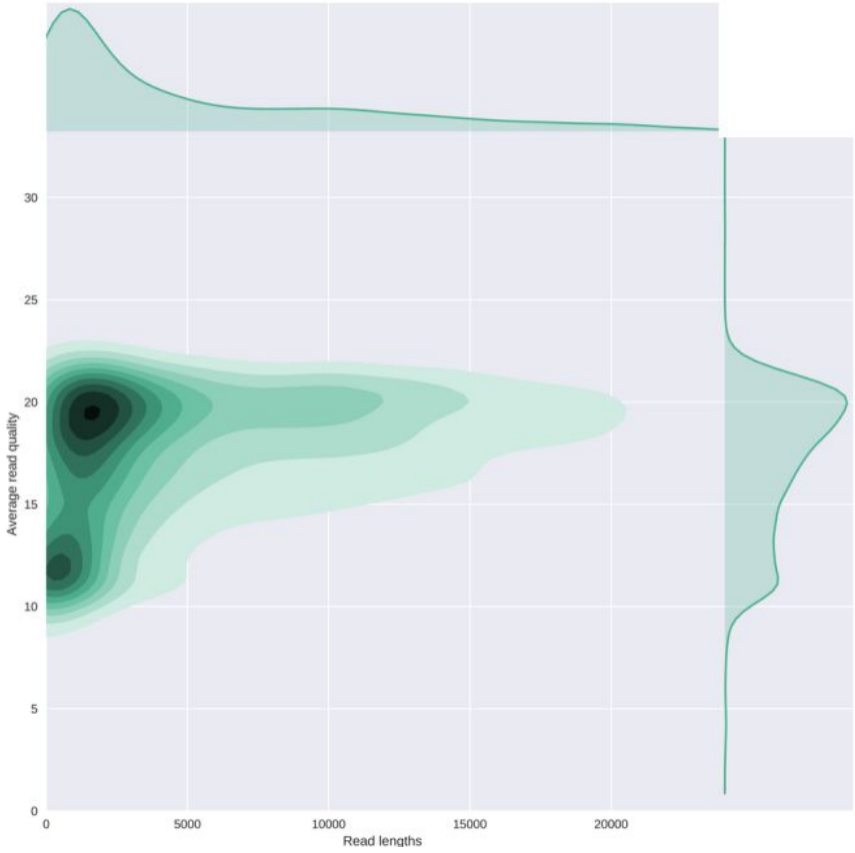

Figure 4.2: Plot from the tool NanoPlot

4.2 Read Processing

Two-step process:

1. Adapter Clipping (long reads and short reads)
2. Quality Trimming (short reads)

4.2.1 Quality trimming for short reads only?

The per base quality for Illumina short reads is mostly very high for the first bases but slowly decreases towards the ends of the reads. Especially with high coverage, trimming a few bases at the ends doesn't lose a substantial amount of information but reduces the error rate and therefore often improves the result of the downstream results.

Long reads on the other hand may have regions of low quality across the whole read. Here, quality trimming could mean splitting the read into multiple fragments, which ultimately contradicts the purpose of long read sequencing. Long reads should only be trimmed or split carefully, some tools do so, but rather conservatively[2]

4.2.2 Adapter clipping or quality trimming first?

Partially trimming adapter sequences might prevent software tools from identifying the remainder of the adapter sequences.

4.2.3 (Some) Tools

Popular tools for short reads are:

- Trimmomatic [3]
- CutAdapt [4]
- Fastp [5]
  (Uses CutAdapt for adapter clipping and Trimmomatic for quality trimming but can detect adapter sequnces automatically)

Popular tools for long reads are:

- Porechop
- Porechop_ABI[1] [6]

4.2.4 Finding your adapter sequences

Illumina adapter portfolio

[1] Regular Porechop uses a database of adapter sequences while Porechop_ABI detects by with k-mer counting

Chapter 5

De Novo Assembly

De novo assembly aims at the in silico reconstruction of large DNA or RNA molecules from the genome or transcriptome from fragments without the need for a reference genome. It is particularly useful when the genome of an organism or the target sequence is unknown or not well-characterized.

5.1 A Naive Algorithm

This naive algorithm successively computes the best possible overlap between pairs of reads and merges them together to form a longer read.

Algorithm 1 Naive Assembly Algorithm

```
procedure NaiveAssemble(L, a list of reads)
    From L remove all strings that are substrings of other reads from L
    while |L| > 1 do
        Compute the best possible possible overlap for each pair of reads in L
        Merge the pair of reads with the best possible overlap
    end while
    return L[0]
end procedure
```

The algorithm is too inefficient for the large data sets of high-throughput methods. The time required to calculate an overlap alignment is quadratic to the read length and we would have to calculate a lot of them ($(\#reads)^2 + \Sigma_{i=1}^{\#reads-1}$). Also it does not address the fact that we might not want a result that is a single sequence since the reads might come from different chromosomes or transcripts.

5.2 Objectives of Assembly Algorithms

- Memory optimization
- Run time optimization
- Consideration and differentiation of sequencing errors and mutations as well as modifications of nucleobases
- Consideration of Gaps
  - Chromosome borders (We want these)
  - 0 coverage (We don't want these
- Consideration of repeats
  - short repeats
  - gene duplication
- Differentiation of + strand and - strand
- Filtering chimeras (from unintentional ligation during preparation)

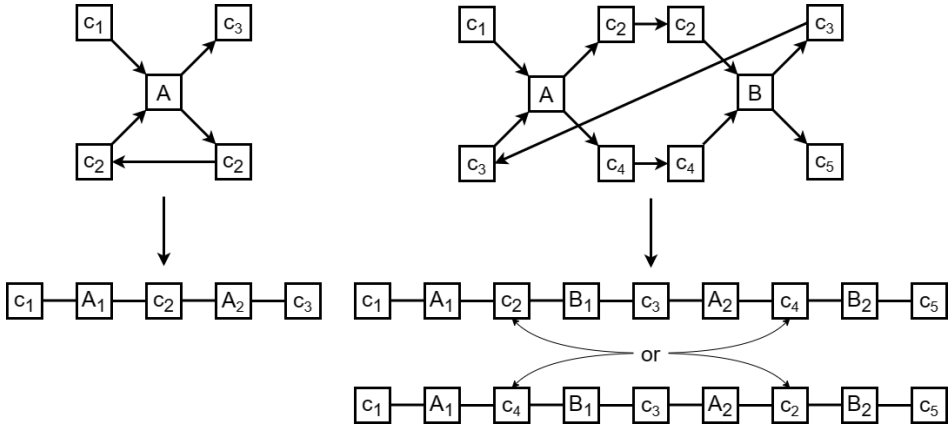

Figure 5.1: Paralogous genes may cause problems for genome assembly

Figure 5.1 demonstrates how duplicated sequences might lead to multiple solutions of the same quality. Only one of those solutions is the biological truth but it is not possible to tell. The Figure shows two examples of overlap graphs (subsection 5.4.1) that are product of genomes with duplicates. For both of them we try to reconstruct the original sequence by traversing the overlap graphs. This leads to a unique solution for the example on the left but multiple for the example on the right.

5.3 Results of an Assembly

5.3.1 Levels of Assembly

1. Reads
2. Contigs
   (Contiguous sequence stretches assembled from reads)
3. Scaffolds
   (Contiguous sequence stretches, with unresolved sequence stretches inbetween connected by using data for example from chromatin conformation capture)

5.3.2 Quality Scores

- Length of the longest
- Number of contigs
- N50 score

  Length of the shortest contig, such that the sum of the lengths of all even shorter sequences make up 50% of the total length of the reads
- L50

  Minimal number of contigs such that their total length makes up 50% of the genome length

Note that all of these scores aren't actually measuring the accuracy of the assembly but only the length of resulting sequences (which could be maximized by simply merging reads without any overlap at all).

5.3.3 BUSCO

Benchmarking Universial Single Copy Orthologs are highly conserved genes that usually can be found exactly once in the genome of each species within a big phylogenetic clade. The BUSCO software checks if these genes were assembled correcly. [7]

5.4 Solving the Shortest Superstring Problem

Given a set of strings $S = \{s_1, ..., s_n\}$ we want to find the shortest string s, that contains all s_i as substrings. This Problem is NP-hard.

5.4.1 Hamilton Paths in Overlap Graphs

Definition 5: Overlap graph

$G = (S, E, \mu)$
$E = S^2$(all ordered pairs of substrings)
$\mu(E) =$ the maximum amount of overlap possible with this ordered pair of strings

A Hamilton path is a path in a graph that contains each node exactly once. Each of these paths corresponds to a way to piece the smaller strings together to form a superstring. A path that corresponds to a merging with the maximum total overlap, leads to the shortest possible superstring. However, finding such paths is NP-hard.

Figure 5.2 shows an example of an overlap graph with eight reads of length three. The optimal Hamilton paths are:
ATG→TGG→GGC→GCG→CGT→GTG→TGC→GCA
and
ATG→TGC→GCG→CGT→GTG→TGG→GGC→GCA.

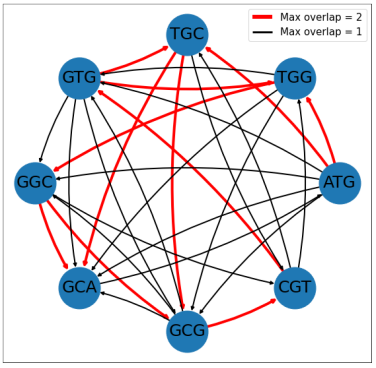

Figure 5.2: Overlap graph

5.4.2 Euler Paths in De Bruijn Graphs

Definition 6: De Bruijn graph

De Brujin graphs are directed graphs defined for a set of symbols and a number $k \in \mathbb{N}$. Let K be the set of k-mers that can be formed of the symbols. The De Bruijn graph then is defined as:

$$G = (K, E, \mu)$$
$$E = \{(k, k') \mid k, k' \in K \text{and} k_2, ...k_{k-1} = k_1, ...k_k\}$$

The vertices represent the k-mers. If one of the k-mers can be expressed as another k-mers by removing the first symbol and adding one at the end, then the corresponding vertices are connected by an edge.

Additionally, we define a weight $\mu : E \to \mathbb{N}$ with $\mu(k, k')$ being the number of times k starts at a position i and k' at position $i + 1$ in a String $s \in S$

An Euler path is a path in a graph that contains each edge e $\mu(e)$ times. Each of these paths corresponds to a way to piece the smaller strings together to form a superstring. Here we are interested in the shortest paths. These paths can be found in linear time and we can limit the size of the graph with the choice of k.

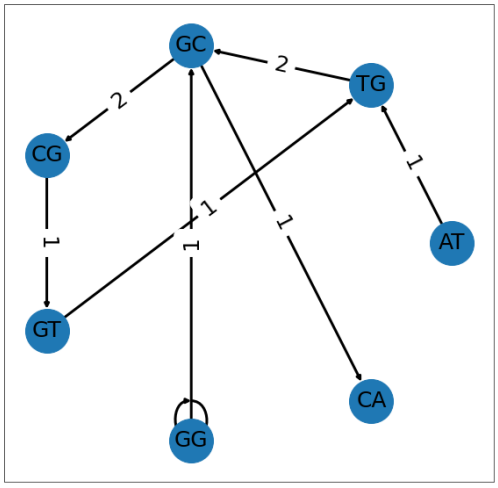

Figure 5.3: De Brujin graph for $S =$ {'ATG', 'TGCG', 'TGC', 'GTG', 'GGGC', 'GCA', 'GCG', 'CGT'}

5.4.3 Comparison Hamilton vs Euler

Both of the approaches have their own advantages and disadvantages. Overlap graphs offer biological intuition and use more information but can be computationally intensive. However, it could be of use to assemble few very long sequences. The Euler approach sacrifices some of the information in favor of reduced complexity and can handle larger amounts of short sequences. Most modern assemblers are variations of these approaches.

Table 5.1: This table, that unfortunately is in the German language since I was to lazy to remake it, compares the expected performance of both the Hamilton and Euler approach in different categories when confronted with the task of genome assembly. ↑ is good, ↓ is not. The categories are: a) required memory, b) speed and dealing with c) sequencing errors and biological variation, d) gaps, e) repeats, f) strand orientation and g) chimeric reads

Problem	Hamilton	Euler
Speicher	↓ $O(\#Reads^2)$	↑ $O(\#Basen^{2k})$
Geschwindigkeit	↓ 1. Bauen des Graphen: Berechnung einer besten Überlappung: $O(Readlänge^2)$ Anzahl Berechnungen: $O(\#Reads^2)$ 2. Suchen im Graphen: NP-schwer	↑ 1. Bauen des Graphen: Berechnung K-mere eines Reads: $O(Readlänge)$ Prüfen ob K-mer bereits Knoten: $O(1)$ Einfügen von Kanten: $O(1)$ 2. Suchen im Graphen: Linearzeit-Algorithmen
Mutationen, Sequenzierfehler, Modifikationen	↑	↓
Lücken	↑ Falsches Schließen von Lücken möglich, wenn Reads zufällig gut überlappen. Weniger wahrscheinlich für lange Reads und hohe Coverage	↓ Falsches Schließen von Lückenwahrscheinlich, da K-mere mit hoher Wahrscheinlichkeit mehrfach vorkommen.
Repeats	↑ Mehr Kontextinformation	↓ Weniger Kontextinformation
Orientierungs-problem	↓ Wird ignoriert	↓ Wird ignoriert
Chimären	↑↓ Wird ignoriert Vermutung: Mögliche Kanten des entsprechenden Knoten gehören häufig nicht zum besten Hamilton-pfad, da sie nicht zu den übrigen Daten „passen“.	↓ Wird ignoriert

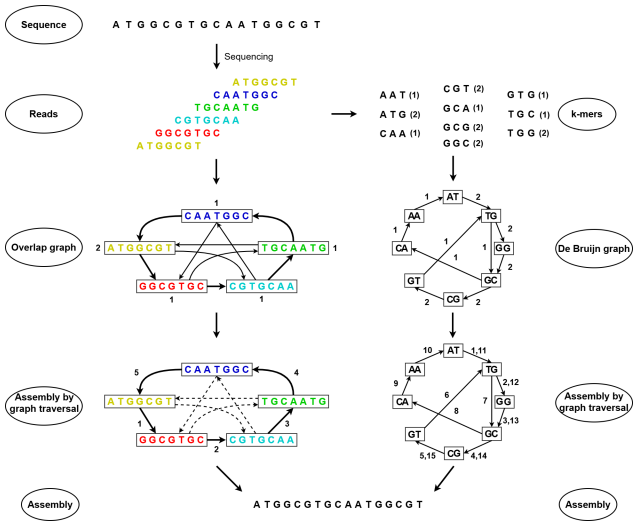

Figure 5.4: Caption

5.5 Velvet assembler

A set of algorithms to manipulate de Bruijn graphs for genomic sequence assembly. It makes use of a modified and more efficient representation of the graph called the assembly graph as shown in Figure 5.5.

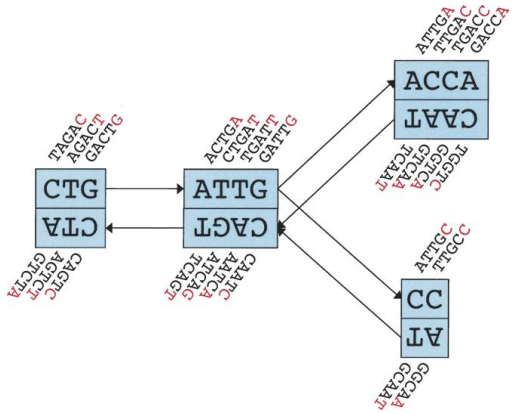

Figure 5.5: From the paper: Schematic representation of our implementation of the de Bruijn graph. Each node, represented by a single rectangle, represents a series of overlapping k-mers (in this case, k = 5), listed directly above or below. (Red) The last nucleotide of each k-mer. The sequence of those final nucleotides, copied in large letters in the rectangle, is the sequence of the node. The twin node, directly attached to the node, either below or above, represents the reverse series of reverse complement k-mers. Arcs are represented as arrows between nodes. The last k-mer of an arc's origin overlaps with the first of its destination. Each arc has a symmetric arc. Note that the two nodes on the left could be merged into one without loss of information, because they form a chain.

The algorithms resolve possible artefacts of sequencing errors and lead to improved runtime by simplifying the graph.

1. Removing tips that represents less than 2k nucleotides
   (A "tip" is a chain of nodes that is disconnected from the graph on one end)

2. Removing bubbles with the tour bus algorithm
   (A bubble is a pair of paths that starts and ends at the same nodes and is thereby redundant)

3. Removing erroneous connections
   (Utilizes a basic coverage cutoff)

[8]

Think of when these simplification might benefit your result and when it might harm them. Were there any biologically significant edges that were removed because they were mistaken for artifacts of sequencing errors?

Try to think of examples for both cases!

Velvet shouldn't be used for genome assembly any longer since there exist much more advanced tools. They are, however based on the ideas first implemented in velvet.

5.6 Short-, Long- and Hybrid-assembly

Genomes and transcriptomes can be reconstructed using short reads, long reads or a combination of both.

5.6.1 Short Read Assemblies

In order to achieve almost complete coverage of the genome or transcriptome with Illumina sequencing, on average each position must be covered multiple times. The total amount of reads usually makes the computation of overlap graphs impossible. Most modern short read assemblers work with assembly graphs similar to the ones produced by velvet (**??**). Paths corresponding to the resulting sequences are then often computed using heuristic path finding algorithms.

5.6.2 Long Read Assemblies

Long reads are much longer and provide more structural information. Therefore, compared to Illumina sequencing a lower coverage is sufficient and efficient mapping algorithms allow the use of overlap approaches. Assemblies from long reads tend to produce much longer contigs but with lower per base accuracy.

5.6.3 Hybrid Assemblies

Hybrid assemblies attempt to leverage the advantages of both short and long read sequencing methods. Some short read assemblers for example were extended to use the long reads to find paths in the assembly graph [9]. A different approach to use the more accurate short reads to correct the errors of the long reads by directly or indirectly mapping them to each other, finding consensus sequences and then performing a long read assembly.

Chapter 6

Read Mapping

6.1 Read Mapping with Burrows-Wheeler

Bowtie is an ultrafast, memory-efficient short read aligner that makes use of the Burrows-Wheeler Transformation. [10]

Algorithm 2 Burrows-Wheeler Transformation

1: **procedure** BWT(s (string))
2: $s\$ \leftarrow s +' \$'$
3: $L \rightarrow [s\$_i, ..., s\$_{|s\$|+s\$_1,...,s\$_{i-1}}\ for\ i\ in\ 1, ..., |s\$|]]$ ▷ Burrows-Wheeler matrix, list of cyclic rotations
4: Sort the strings in L lexicographically
5: $s' \leftarrow L_1[|s|], ..., L_{|s|}[|s|]$ ▷ Last column of the Burrows-Wheeler matrix
6: **return** s'
7: **end procedure**

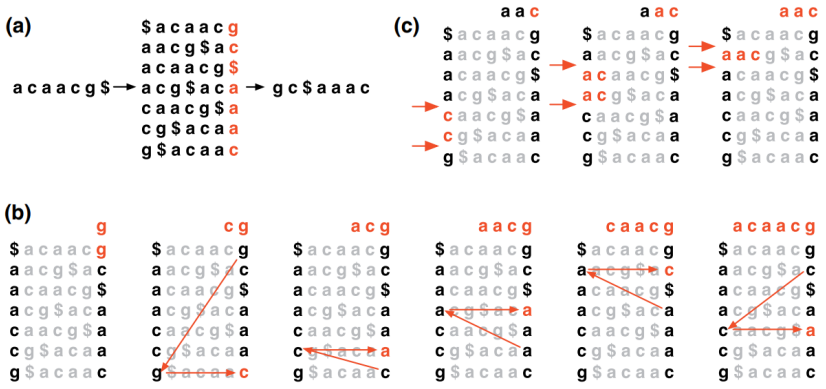

Figure 6.1: Burrows-Wheeler transform. **(a)** The Burrows-Wheeler matrix and transformation for 'acaacg'. **(b)** Steps taken by EXACTMATCH to identify the range of rows, and thus the set of reference suffixes, prefixed by 'aac'. **(c)** UNPERMUTE repeatedly applies the last first (LF) mapping to recover the original text (in red on the top line) from the Burrows-Wheeler transform (in black in the rightmost column).

Let's take a closer look at the Burrows-Wheeler matrix (the lexicographically sorted list of rotated strings)! Because of the way matrix was generated it is quite obvious that:

1. The strings are grouped by their initial letter (because of the lexicographic sorting).
2. In each of the rotated strings, the first letter is the sucessor of the last letter in the original string (because of the rotating).

What is maybe a bit less obvious:

1. This matrix has a property called 'last first (LF) mapping. That is that "the i-th occurrence of character X in the last column corresponds to the same text character as the i th occurrence of X in the first column" and both occurrences correspond to the same position in the original String.

We can simply determine these i for both the first and last column by iterating over the columns and count the occurrences for each character!

How does this help us with the exact mapping of reads?

Aligning in Bowtie is an iterative process where we start with aligning the last character of a read.

That means mapping a read $r = r_0, r_1, ..., r_m$ to a genome $g = g_0, g_1, ..., g_n$ requires us to map the read $r' = r_1, ..., r_m$ to the genome first. So in every step, after we found all i such that $r' = r_1, ..., r_m$ can be aligned to $g_{i+1}, ..., g_{i+m}$ (mapping of r') all that is left to do is checking for every i if $r_0 = g_i$ (mapping of r by alignment extension).

Each character of the genome we mapped to in the previous iteration step corresponds to a row in the Burrow-Wheeler matrix. For the selected rows we determine those that end with the character that is to be aligned next.

The Burrows-Wheeler matrix has $O(|genome|^2)$ elements. Consider the chromosomal DNA of E.coli which is only $4,600kb$ long, so the Burrows-Wheeler matrix would have $21,160,000,000,000$ elements already.

But only the first and the last column of the matrix are actually needed in the alignment steps. Thus, all the columns in-between don't have to be generated in the first place. Additionally, since the first column is grouped by characters a list with the indices where the groups start or end is as good as the actual first column but has only 5 integer numbers (one for each nucleobase and one for the'$').

The mapping step in more more detail from the paper:

In a series of steps, the EXACTMATCH algorithm (Figure 6.1) calculates the range of matrix rows beginning with successively longer suffixes of the query. At each step, the size of the range either shrinks or remains the same. When the algorithm completes, rows beginning with S0 (the entire query) correspond to exact occurrences of the query in the text. If the range is empty, the text does not contain the query. UNPERMUTE is attributable to Burrows and Wheeler and EXACTMATCH to Ferragina and Manzini.

Figure 6.2: Example of a read mapping from the original Bowtie paper

EXACTMATCH is insufficient for short read alignment because alignments may contain mismatches, which may be due to sequencing errors, genuine differences between reference and query organisms, or both. We introduce an alignment algorithm that conducts a backtracking search to quickly find alignments that satisfy a specified alignment policy. Each character in a read has a numeric quality value, with lower values indicating a higher likelihood of a sequencing error. Our alignment policy allows a limited number of mismatches and prefers alignments where the sum of the quality values at all mismatched positions is low.

The search proceeds similarly to EXACTMATCH, calculating matrix ranges for successively longer query suffixes. If the range becomes empty (a suffix does not occur in the text), then the algorithm may select an already-matched query position and substitute a different base there, introducing a mismatch into the alignment. The EXACTMATCH search resumes from just after the substituted position. The algorithm selects only those substitutions that are consistent with the alignment policy and which yield a modified suffix that occurs at least once in the text. If there are multiple candidate substitution positions, then the algorithm greedily selects a position with a minimal quality value.

Backtracking scenarios play out within the context of a stack structure that grows when a new substitution is introduced and shrinks when the aligner rejects all candidate alignments for the substitutions currently on the stack. See Figure 6.2 for an illustration of how the search might proceed.

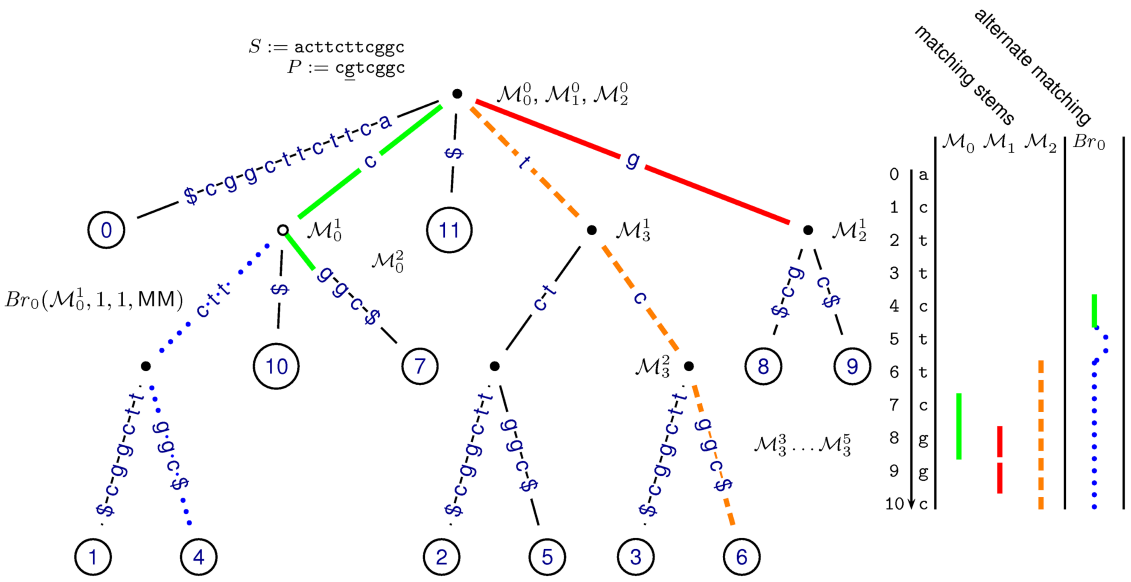

Figure 6.3: Example from the original SEGEMEHL paper

6.2 Read Mapping with Indexing Structures

SEGEMEHL is a mapping tool, that can not only deal with mismatches but with a limited number of insertions and deletions as well. Genomic reference sequences in SEGEMEHL are stored as suffix interval trees which are efficient representations for suffix trees. [11]

Definition 7: Suffix tree

A suffix tree T of a string S is the smallest possible tree were edges represent substrings of S and each suffix of S is represented by one and only one path from the root.

- It has |S| leaves
- Each inner node has at least two children
- Substrings that are part of different suffixes appear only once

Chapter 7

Differential Gene Expression Analysis

Comparing gene expression levels between different different conditions, such as healthy vs diseased.

For each gene we are interested in two things:

- The magnitude of the change in expression levels
- The statistical significance of this change

7.1 Read Counting

Measuring gene expression levels is done in three steps

1. Mapping reads to an annotated reference genome
2. Counting the number of aligned reads that were aligned for each gene
3. Normalizing the read counts

Use featureCounts [12] for read counting.

7.1.1 Multiple Mapping Problem

What if reads align to multiple genes? How do we count them?

- Unique → Increment the count for only one gene the read maps to by 1
- Multiple → Increment the count for all genes the read maps to by 1
- Fraction → Increment the count for all genes the read maps to by 1/#(genes the read maps to)

7.1.2 Normalization of Read Counts

Why we normalize read counts:
Instead of looking at absolute differences in gene expression, we look at relative differences. Some samples could contain more or larger cells than others and therefore have more transcripts, without being in a biologically different condition.

Definition 8: RPKM

$$RPKM = \frac{reads\ mapped\ to\ gene}{(gene\ length\ in\ kb) * \frac{(total\ number\ of\ reads)}{10^6}}$$

Definition 9: FPKM

$$FPKM = \frac{fragments\ mapped\ to\ gene}{(gene\ length\ in\ kb) * \frac{(total\ number\ of\ fragments)}{1{,}000{,}000}}$$

Definition 10: TPM

$$TPM = \frac{RPKM}{\sum(RPKMs)} * 10^6$$

Watch: RPKM, FPKM and TPM, Clearly Explained!!!

TPM is the mighty mouse of read count normalization method, use it!

7.2 Identifying statistically significant changes

7.2.1 p-Values

p-Values indicate statistical significance.

In null-hypothesis testing the probabilities of different results of an experiment are assumed to be randomly (often uniformly) distributed. The p-Value expresses the probability to observe a specific result or a result that is even more unlikely to happen. A small p-value might indicate that the null-hypothesis is to be rejected and the outcome of the experiment is not random or follows a different random distribution than initially assumed.

Some good videos on p-values by StatsQuest:

p-values: What they are and how to interpret them (11:21)
How to calculate p-values (25:15)
StatQuest: One or Two Tailed P-Values (7:05)

What we are interested in: Are the measured differences in gene expression levels meaningful or due to chance?

Example

Throw a coin ten times. We would usually assume that for both heads and tails there is a probability of 0.5 to observe them and thus expect to observe heads five times and tails five times. Would you be surprised to see heads six times? Would you be surprised to see heads nine times?

$$p(\#heads \geq x) = \sum_{k=x}^{10} \binom{10}{k} * 0.5^k * 0.5^{(n-k)} \tag{7.1}$$

$$p(\#heads \geq 6) \approx 0.38 \tag{7.2}$$

$$p(\#heads \geq 9) \approx 0.01 \tag{7.3}$$

Six or more heads are quite likely, nine or more heads is very unlikely. Was the coin manipulated to increase someones betting luck?

7.2.2 Corrected p-values

Repeating experiments that are subject to randomness will eventually lead to the observation of rare events and p-values should be adjusted according to the number of times the experiment is repeated. This is very important for differential gene expression analysis since we are usually looking at thousands of genes simultaneously.

Example

Throw a fair coin ten times. The probability to observe heads ten times is only $p(\#heads = 10) = 0.5^{10} \approx 0.001$ Repeat this every day and the probability of never observing heads ten times on a single time shrinks to $p(no\ 10\ heads\ in\ 20\ years) \approx (1 - p(\#heads = 10)^{20*365} \approx 0.01$

7.2.3 The Null-hypothesis for Gene Expression Levels

Poisson Distribution and Overdispersion

Definition 11: Poisson distribution

A discrete probability distribution that expresses the probability of a given number of events occurring in a fixed interval of time if these events occur with a known constant mean rate and independently of the time since the last event. (wikipedia)

$$P(k) = \frac{\lambda^k e^(-\lambda)}{k!}$$

Mean: λ
Variance: λ

The definition of this distribution reads like it is exactly what we need: A gene is ready for transcription and is read by the RNA polymerase at an average rate of lambda. λ is both the mean value and the variance of the Poisson distribution. However, empirically we observe that the variance for gene expression levels is higher than it's mean. This phenomenon is called overdispersion.

Negative Binomial Distribution

To accurately describe gene expression levels with a statistical model a distribution were the variance can be controlled by a separate parameter is needed. The negative binomial distribution is the model of choice.

Definition 12: Negative binomial distribution

a discrete probability distribution that models the number of failures in a sequence of independent and identically distributed Bernoulli trials before a specified (non-random) number of successes r

$$P(X = k) = \binom{k + r - 1}{k} p^r (1 - p)^k$$

Mean: $\frac{r(1-p)}{p}$
Variance: $\frac{r(1-p)}{p^2}$

Video explaining regression, poisson regression, overdispersion and negative binomial regression (19:35)

7.2.4 Vulcano Plots

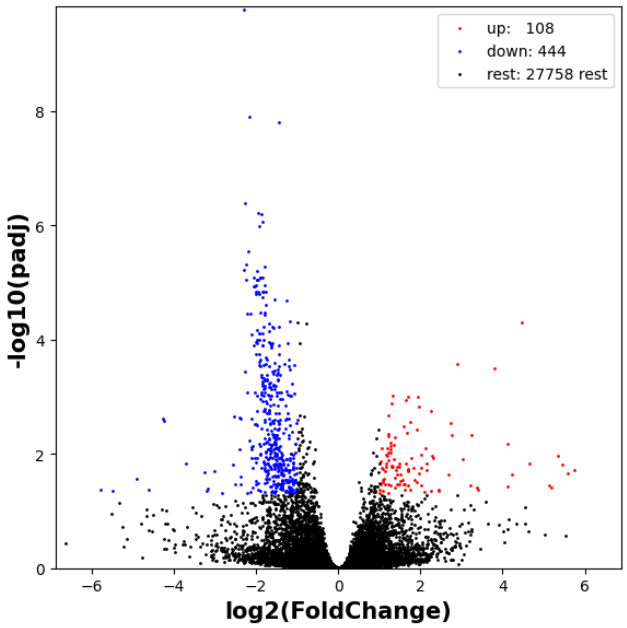

Figure 7.1: Exemplary vulcano-plot

7.3 Tools

- DESeq2 [13]
- EdgeR [14]

DESeq2 is probably the most popular tool and implemented in R. There is, however, a re-implementation in Python for anyone that is not secretly a lizard.


So uncivilized.

Chapter 8

File Formats

8.1 Fasta/Fastq

Fasta is a text-based format for storing raw reads.

Table 8.1: Fields in a Fasta-file

	Content	**Explanation**
1	@SEQ_ID	ID that usually also informs about the flow cell and flowcell position of the read
2	Sequence	The sequence of the read

Basically fasta with two additional lines.

Table 8.2: Fields in a Fastq-file

	Content	**Explanation**
1	@SEQ_ID	ID that usually also informs about the flow cell and flowcell position of the read
2	Sequence	The sequence of the read
3	+	Begins with a plus and is usually empty
4	Confidence	AscII encoded Phred score[a]

[a] There is a bijective map between the AscII-characters and the numbers $1, ..., 255$. Each character represents a Phred quality score of it's corresponding number minus 33 (Sanger standart) [15]

8.2 BAM/SAM

Sequence Alignment Map (SAM) is a text-based format for storing biological sequences aligned to a reference sequence.

Table 8.3: Fields in a SAM-file

	Name	Type	Meaning
1	QNAME	String	Query template NAME
2	FLAG	Int	Bit wise FLAG (properties of the mapping)
3	RNAME	String	References sequence NAME
4	POS	Int	1- based leftmost mapping POSition
5	MAPQ	Int	MAPping Quality
6	CIGAR	String	CIGAR string
7	RNEXT	String	Ref. name of the mate/next read
8	PNEXT	Int	Position of the mate/next read
9	TLEN	Int	Observed Template LENgth
10	SEQ	String	Segment SEQuence
11	QUAL	String	QUALity

BAM-files are the binary equivalents of SAM-files and need less memory.

8.2.1 CIGAR - String

The CIGAR - String encodes how the feature is aligned to the reference (Example: Exons of a gene vs a reference genome)

- $x\mathbf{M}$: Match of length x
- $x\mathbf{X}$: Mismatch of length x
- $x\mathbf{D}$: Gap in the feature of length x
- $x\mathbf{I}$: Gap in the reference of length x

Example:

Alignment:

A	G	C	A	T	T	C	G	C	A	C	A
A	T	C	A	-	-	C	G	C	A	C	A

Corresponding CIGAR string: 1M1X2M2D4M

(1 Match, 1 Mismatch, 2 Matches, 2 Deletions, 4 Matches)

8.3 GFF (Genomic Feature Format)

GFF is a text-based format for storing genomic features. Each line represents a feature and has multiple fields.

Table 8.4: Fields in a Genomic Feature Format file

	Name	**Meaning**	**Example**
1	seqid	location	chr1
2	source	algorithm or database	BLAST
3	type	feature type name	gene
4	start	start of the feature, with a 1-base offset	136532
5	end	end of the feature, with a 1-base offset	278667
6	score	confidence	0.00342
7	strand	strand of the feature	"+", "-", "." (unknown), or "?" (relevant but unknown)
8	phase	0, 1, 2 (for CDS features) or "."	1
9	attributes	additional information	name=loyalty gene;origin=Kamino

Bibliography

[1] Yunhao Wang, Yue Zhao, Audrey Bollas, Yuru Wang, and Kin Fai Au. Nanopore sequencing technology, bioinformatics and applications. *Nature biotechnology*, 39(11):1348–1365, 2021.

[2] Leandro Lima, Camille Marchet, Ségolène Caboche, Corinne Da Silva, Benjamin Istace, Jean-Marc Aury, Hélène Touzet, and Rayan Chikhi. Comparative assessment of long-read error correction software applied to nanopore rna-sequencing data. *Briefings in bioinformatics*, 21(4):1164–1181, 2020.

[3] Anthony M Bolger, Marc Lohse, and Bjoern Usadel. Trimmomatic: a flexible trimmer for illumina sequence data. *Bioinformatics*, 30(15):2114–2120, 2014.

[4] Marcel Martin. Cutadapt removes adapter sequences from high-throughput sequencing reads. *EMBnet. journal*, 17(1):10–12, 2011.

[5] Shifu Chen, Yanqing Zhou, Yaru Chen, and Jia Gu. fastp: an ultra-fast all-in-one fastq preprocessor. *Bioinformatics*, 34(17):i884–i890, 2018.

[6] Quentin Bonenfant, Laurent Noé, and Hélène Touzet. Porechop_abi: discovering unknown adapters in oxford nanopore technology sequencing reads for downstream trimming. *Bioinformatics Advances*, 3(1):vbac085, 2023.

[7] Felipe A Simão, Robert M Waterhouse, Panagiotis Ioannidis, Evgenia V Kriventseva, and Evgeny M Zdobnov. Busco: assessing genome assembly and annotation completeness with single-copy orthologs. *Bioinformatics*, 31(19):3210–3212, 2015.

[8] Daniel R Zerbino and Ewan Birney. Velvet: algorithms for de novo short read assembly using de bruijn graphs. *Genome research*, 18(5):821–829, 2008.

[9] Dmitry Antipov, Anton Korobeynikov, Jeffrey S McLean, and Pavel A Pevzner. hybridspades: an algorithm for hybrid assembly of short and long reads. *Bioinformatics*, 32(7):1009–1015, 2016.

[10] Ben Langmead, Cole Trapnell, Mihai Pop, and Steven L Salzberg. Ultrafast and memory-efficient alignment of short dna sequences to the human genome. *Genome biology*, 10(3):1–10, 2009.

[11] Steve Hoffmann, Christian Otto, Stefan Kurtz, Cynthia M Sharma, Philipp Khaitovich, Jörg Vogel, Peter F Stadler, and Jörg Hackermüller. Fast mapping of short sequences with mismatches, insertions and deletions using index structures. *PLoS computational biology*, 5(9):e1000502, 2009.

[12] Yang Liao, Gordon K Smyth, and Wei Shi. featurecounts: an efficient general purpose program for assigning sequence reads to genomic features. *Bioinformatics*, 30(7):923–930, 2014.

[13] Michael I Love, Wolfgang Huber, and Simon Anders. Moderated estimation of fold change and dispersion for rna-seq data with deseq2. *Genome biology*, 15:1–21, 2014.

[14] Mark D Robinson, Davis J McCarthy, and Gordon K Smyth. edger: a bioconductor package for differential expression analysis of digital gene expression data. *bioinformatics*, 26(1):139–140, 2010.

[15] Peter J. A. Cock, Christopher J. Fields, Naohisa Goto, Michael L. Heuer, and Peter M. Rice. The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. *Nucleic Acids Research*, 38(6):1767–1771, 12 2009.

List of Tables

List of Figures