Alignment

9

Melanie Kappelmann-Fenzl

Contents

What You Will Learn in This Chapter

The purposes of the alignment process are to measure distances/similarities between strings and thus to locate origins of Next Generation Sequencing (NGS) reads in a reference genome. Alignment algorithms like BLAST that can be used to search for the location of a single or a small number of sequences in a certain genome are not

(continued)

M. Kappelmann-Fenzl (✉)
Deggendorf Institute of Technology, Deggendorf, Germany

Institute of Biochemistry (Emil-Fischer Center), Friedrich-Alexander University Erlangen-Nürnberg, Erlangen, Germany
e-mail: melanie.kappelmann-fenzl@th-deg.de

M. Kappelmann-Fenzl (ed.), *Next Generation Sequencing and Data Analysis*, Learning Materials in Biosciences, https://doi.org/10.1007/978-3-030-62490-3_9

suitable to align millions of NGS reads. This led to the development of advanced algorithms that can meet this task, allow distinguishing polymorphisms from mutations and sequencing errors from true sequence deviations. For a basic understanding, the differences between global and local alignment and the underlying algorithms are described in a simplified way in this chapter, as well as the main difference between BLAST and NGS alignment is described in a simplified way in this chapter. Moreover, different alignment tools and their basic usage are presented, which enables the reader to perform and understand alignment processes of sequencing reads to any genome using the respective commands.

9.1 Introduction

Sequence Alignment is a crucial step of the downstream analysis of NGS data, where millions of sequenced DNA fragments (reads) have to be aligned with a selected reference sequence within a reasonable time. However, the problem here is to find the correct position in the reference genome from where the read originates. Due to the repetitive regions of the genome and the limited length of the reads ranging from 50 to 150 bp, it often happens that shorter reads can map at several locations in the genome. On the other hand, a certain degree of flexibility for differences to the reference genome must be allowed during alignment in order to identify point mutations and other genetic changes.

Due to the massive amount of data generated during NGS analyses, all alignment algorithms use additional data structures (indices) that allow fast access and matching of sequence data. These indices are generated either over all generated reads or over the entire reference genome, depending on the used algorithm. Algorithms from computer science like hash tables or methods from data compression like suffix arrays are popularly implemented in the alignment tools. With the help of these algorithms, it is possible, for example, to compare over 100 GB of sequence data from NGS analyses with the human reference genome in just a few hours. With the help of high parallelization of computing capacity (CPUs), it is possible to reduce this time significantly. Thus, even large amounts of sequencing data from whole-exome or whole-genome sequencing can be efficiently processed.

9.2 Alignment Definition

The next step is the alignment of your sequencing reads to a reference genome or transcriptome. The main problem you have to keep in mind is that the human genome is really big and it is complex too. Sequencers are able to produce billions of reads per run and are prone to errors. Thus, an accurate alignment is a time-consuming process.

Fig. 9.1 Alignment definition. Sequence alignment is a way of arranging the sequences of DNA, RNA, or protein to identify regions of similarity. The basic principle is comparable to a puzzle (left). An optimal alignment means, an alignment with minimal errors like deletions, insertions, or mismatches—no error is defined as a match (right)

Sequence databases like GenBank (http://www.ncbi.nlm.nih.gov/genbank/) grew rapidly in the 1980s, and thus performing a full dynamic programming comparison of any query sequence to every known sequence soon became computationally very costly. Consequently, the alignment of a query sequence against a database motivated the development of a heuristic algorithm [1], which was implemented in the FASTA program suite [2]. The basic principle of this algorithm is to exclude large parts of the database from the expensive dynamic programming comparison by quickly identifying candidate sequences that share short sections (k-tuples) of very similar sequence with the query. FASTA was then followed by the BLAST program [3], with additional speed advantages and a new feature, which estimates the statistical likelihood that each matching sequence had been found by chance. BLAST is still one of the most used search program for biological sequence databases [4]. With the introduction of ultra-high-throughput sequencing technologies in 2007, other alignment challenges emerged. This chapter describes these efforts and the current state of the art in NGS alignment algorithms. Computational biologists have developed more than 70 read mapping to date [5]. A full list of sequence alignment software tools can be found at https://en.wikipedia.org/wiki/List_of_sequence_alignment_software#Short-Read_Sequence_Alignment. Actually, describing all of these tools is beyond the scope of this chapter, however main algorithmic strategies of these tools are depicted below.

Sequence alignment (Fig. 9.1) is widely used in molecular biology to find similar DNA, RNA, or protein sequences. These algorithms generally fall into two categories: global (Needleman–Wunsch), which aligns the entire sequence, and local (Smith–Waterman), which only look for highly similar subsequences.

9.2.1 Global Alignment (Needleman–Wunsch Algorithm)

Statistically the space for possible solutions is huge; however, we are interested in optimal alignments with minimal errors like indels or mismatches. The so-called unit edit distance (edist) is the number of mismatches, insertions, and deletions in an optimal sequence alignment. The main aim is to minimize the edist by tabulating partial solutions in a (m

$$E(i,j) = \min \begin{cases} E(i-1,j)+1 & \text{Deletion} \\ E(i,j-1)+1 & \text{Insertion} \\ E(i-1,j-1)+1 & \text{Substitution} \end{cases} \qquad E(i,j) = \max \begin{cases} E(i-1,j)-1 & \text{Deletion} \\ E(i,j-1)-1 & \text{Insertion} \\ E(i-1,j-1)-1 & \text{Substitution} \end{cases}$$

Fig. 9.2 Needleman–Wunsch. Optimization of distance (left) and optimization of similarity (right)

	Needleman-Wunsch	
	Optimization of distance	Optimization of similarity
Match	0	0
Mismatch	1	-1
Gap	1	-1
Diagonal jump	Match/ Mismatch	
Top/ Down jump	Deletion	
Left/ Right jump	Insertion	

E	i	G	C	A	C	T
j	0	1	2	3	4	5
T	1	1	2	3	4	4
G	2	1	2	3	4	5
A	3	2	2	2	3	4
T	4	3	3	3	3	3
A	5	4	4	3	4	4
T	6	5	5	4	4	4

T G A T A T
  * | | | *
_ G C A C T

Fig. 9.3 Needleman–Wunsch Algorithm and the resulting Scoring Matrix (E). Matches are defined as 0, Mismatches and Gaps as 1/−1. The *edist* is marked in red [4]. A possible traceback is depicted by blue arrows and the corresponding alignment at the bottom right. Diagonal jumps within the scoring Matrix can be interpreted as Matches or Mismatches, Top or Down jumps as Deletions, and Left or Right jumps as Insertions

+1) x (n+1) matrix. Under the assumption that both input sequences a and b stem from the same origin, a global alignment tries to identify matching parts and the changes needed to transfer one sequence into the other. The changes are scored and an optimal set of changes is identified, which defines an alignment. The dynamic programming approach tabularizes optimal subsolutions in matrix E, where an entry E (i,j) represents the best score for aligning the prefixes $a_{1..i}$ with $b_{1..j}$ (Fig. 9.2).

Scoring Matrix using Needleman–Wunsch algorithm [6] and the corresponding traceback Matrix lead to the identification of the best alignment. One possible alignment result of our example and the related traceback are illustrated in Fig. 9.3.

9.2.2 Local Alignment (Smith–Waterman Algorithm)

Local alignment performed by the Smith–Waterman algorithm [7] aims to determine similarities between two nucleic acid or protein sequences. The main difference to the global alignment is that negative scoring matrix cells are set to zero (Fig. 9.5).

For a better understanding of local/sub-regions alignment imagine you have a little Toy genome (16 bp): CATGGTCATTGGTTCC.

Local alignment is a hash-based algorithm with two major approaches: hashing the reference and the Burrows–Wheeler transform [8, 9]. The first step is to hash/index the genome (forward strand only) resulting in a hash/k-mer index of your Toy genome:

k=3	K-mer/Hash	Positions
	CAT	1,7
	ATG	2
	TGG	3,10
	GGT	4,11
	GTC	5
	TCA	6
	ATT	8
	TTG	9
	GTT	12
	TTC	13
	TCC	14

Now, you want to align a Toy sequencing read (TGGTCA) to this indexed Toy genome. The k-mer index can be used to quickly find candidate alignment locations in the reference genome. For example, the k-mer TGG is assigned to Positions 3 and 10 and the k-mer TCA to position 6. Thus, Burrows–Wheeler transform is just another way of doing exact matches on hashes and check against genome and calculate a score.

This approach tries to identify the most similar subsequences that maximize the scoring of their matching parts and the changes needed to transfer one subsequence into the other. The dynamic programming approach tabularizes optimal subsolutions in matrix E (Fig. 9.4), where an entry $E_{i,j}$ represents the maximal similarity score for any local alignment of the (sub)prefixes $a_{x..i}$ with $b_{y..j}$, where $x,y>0$ are so far unknown and have to be identified via traceback (Fig. 9.5). Note: consecutive gap (Indels) scoring is done linearly.

Alignment to a reference genome can be performed with single- or paired-end sequencing reads, depending on your experiment and library preparation. Paired-end sequencing is recommended for RNA-Seq experiments.

Furthermore, we differ between two types of aligners:

- Splice unaware
- Splice aware

$$E(i,j) = \max \begin{cases} E(i-1,j)-1 & \text{Deletion} \\ E(i,j-1)-1 & \text{Insertion} \\ E(i-1,j-1)-1 & \text{Substitution} \end{cases}$$

Fig. 9.4 Smith–Waterman. Optimization of similarity

	Smith-Waterman
Match	2
Mismatch	-1
Gap	-1
Diagonal jump	Match/ Mismatch
Top/ Down jump	Deletion
Left/ Right jump	Insertion

E	i	G	C	A	C	T
j	0	0	0	0	0	0
T	0	0	0	0	0	2
G	0	2	1	0	0	1
A	0	1	1	3	2	1
T	0	0	0	2	2	4
A	0	0	0	2	1	3
T	0	0	0	1	1	3

G _ A _ T
* * *
G C A C T

Fig. 9.5 Smith–Waterman Algorithm and the resulting Scoring Matrix (E). Matches are defined as 2, Mismatches and Gaps as −1. The traceback is depicted by blue arrows and the corresponding alignment at the bottom right. Diagonal jumps within the scoring Matrix can be interpreted as Matches or Mismatches, Top or Down jumps as Deletions, and Left or Right jumps as Insertions

Table 9.1 Splice-aware and splice-unaware alignment tools

Alignment tool	Splice-aware	Link
STAR	Yes	https://github.com/alexdobin/STAR
Bowtie	No	http://bowtie-bio.sourceforge.net/index.shtml
Bowtie2	Yes	http://bowtie-bio.sourceforge.net/bowtie2/index.shtml
TopHat/TopHat2	Yes	http://ccb.jhu.edu/software/tophat/index.shtml
BWA-MEM	Yes	http://bio-bwa.sourceforge.net/
BWA-SW	No	
BWA-backtrack	No	
Hisat2	Yes	https://ccb.jhu.edu/software/hisat2/manual.shtml
Segemehl	Yes	https://www.bioinf.uni-leipzig.de/Software/segemehl/

Splice-unaware aligners are able to align continuous reads to a reference genome, but are not aware of exon/intron junctions. Hence, in RNA-sequencing, splice-unaware aligners are no proper tool to analyze the expression of known genes, or align reads to the transcriptome. Splice-aware aligners map reads over exon/intron junctions and are

therefore used for discovering new splice forms, along with the analysis of gene expression levels (Table 9.1).

In this context the most common alignment tools are explained in the following section.

9.2.3 Alignment Tools

9.2.3.1 STAR

Spliced Transcripts Alignment to a Reference (STAR) is a standalone software that uses sequential maximum mappable seed search followed by seed clustering and stitching to align RNA-Seq reads. It is able to detect canonical junctions, non-canonical splices, and chimeric transcripts.

The main advantages of STAR are its high speed, exactness, and efficiency. STAR is implemented as a standalone C++ code and is freely available on GitHub (https://github.com/alexdobin/STAR/releases) [10].

In terms of mapping multiple samples, you can parallelize your mapping command. First, create a .txt file containing your file names:

```
parallel -j 1 echo {1} ::: Sample1 Sample2 Sample3 ::: > /path/to/SampleNames.txt
```

You can use the generated *SampleNames.txt* file to combine the commands for mapping and sorting and run it on various samples:

```
cat /path/to/SampleNames.txt | parallel -j 1 "add your mapping commands here
and write "{}" whenever the sample name appears within your commands"
```

Example- Mapping command via STAR (RNA-Seq, paired-end):

```
cat /path/to/SampleNames.txt | parallel -j 1 "mkdir /path/to/mapping_hg38/
{} ; cd /path/to/mapping_hg38/{} ; STAR –runThreadN 15 –genomeDir /path/to/
GenomeIndices/Star/GRCh38_index_100 –readFilesIn /path/to/rawData/{}_r1.
fastq /path/to/rawData/{}_r2.fastq –outFilterType BySJout –outFilter
MultimapNmax 20 –outFilterIntronMotifs RemoveNoncanonicalUnannotated –
outReadsUnmapped Fastq –alignSJoverhangMin 8 –alignSJDBoverhangMin 1 –
alignMatesGapMax 1000000 –alignIntronMax 1000000 –outSAMtype BAM
SortedByCoordinate –quantMode GeneCounts –outWigType bedGraph –
outWigStrand Stranded"
```

The mapping job can be checked in the *Log.progress.out* file in the run directory. This file is updated every minute and shows the number of reads that have been processed, and various mapping statistics. This is useful for initial quality control during the mapping job.

Log.final.out contains the summary mapping statistics of the run.

In the next step, the bedgraph files are sorted and converted to bigwig files (bedGraphToBigWig).

Example:

```
cat /path/to/SampleNames.txt | parallel -j 1 "LC_COLLATE=C sort -k1,1 -k2,2n
/path/to/ /mapping_hg38/{}/Signal.Unique.str1.out.bg > /path/to/
mapping_hg38/{}/Signal.Unique.str1.out.sorted.bg ; bedGraphToBigWig /
path/to/mapping_hg38/{}/Signal.Unique.str1.out.sorted.bg /path/to/
GenomeIndices/Star/GRCh38_index_100/GRCh38.chromosome.sizes /path/to/
bigWig/{}.Signal.Unique.str1.out.bigWig;
```

9.2.3.2 Bowtie

Bowtie is an ultrafast and memory-efficient short read alignment tool to large reference genomes indexed with a Burrows–Wheeler index. It is typically used for aligning DNA sequences as it is a splice-unaware tool. Because of this feature this tool is often used in microbiome alignment (http://bowtie-bio.sourceforge.net/index.shtml) [11, 12].

```
bowtie [options]* -x <ebwt> {-1 <m1> -2 <m2> | -12 <r> | -interleaved <i> | <s>}
[<hit>]
```

Explanation	
-x	path to *Bowtie* index
-q	query input files are FASTQ
-1 <m1>	fastq input files (first mate)
-2 <m2>	fastq input files (second mate)
-c <s>	unpaired reads
--12 <r>	mixed sequencing reads (unpaired and paired-end)
--interleaved <i>	interleaved paired-end FASTQ files

9.2.3.3 Bowtie2

Bowtie2, as well as *Bowtie*, is an ultrafast and memory-efficient tool, but more suitable for aligning sequencing reads of about 50 up to 100s or 1,000s of characters to relatively long reference sequences (e.g., mammalian genomes) indexed with an Ferragina–Manzini (Fm) index. *Bowtie2* supports gapped, local, and paired-end alignment modes (https://github.com/BenLangmead/bowtie2) [13].

```
#Single-end sequencing reads
bowtie2 -x /path/to/GenomeIndices/bowtie2/GRCh38 -U *.fq
#Paired-end sequencing reads
bowtie2 -x /path/to/GenomeIndices/bowtie2/GRCh38 -1 *.1_fq -2 *.2_fq
```

Explanation	
-x	path to *Bowtie2* index
-U <r>	unpaired reads
-1 <m1>	fastq input files (first mate)
-2 <m2>	fastq input files (second mate)

9.2.3.4 TopHat/TopHat2

TopHat aligns RNA-Seq reads to genomes by first using the short-read aligner *Bowtie*, and then by mapping to a reference genome to discover RNA splice sites *de novo*. RNA-Seq reads are mapped against the whole reference genome, and those reads that do not map are set aside. TopHat is often paired with the software Cufflinks for a full analysis of sequencing data (https://github.com/dnanexus/tophat_cufflinks_RNA-Seq/tree/master/tophat2) [14].

A detailed description of the usage of TopHat can be found in the TopHat manual (http://ccb.jhu.edu/software/tophat/manual.shtml).

```
tophat [options]* <genome_index_base> <reads1_1[,...,readsN_1]>
[reads1_2,...readsN_2]
```

Explanation	
<genome_index_base>	path to *TopHat* index
<reads1_1>	fastq input files (unpaired or first mate if paired-end)
<reads1_2>	fastq input files (second mate if paired-end)

9.2.3.5 Burrow–Wheeler Aligner (BWA)

BWA is a splice-unaware software package for mapping low-divergent sequences against a large reference genome, such as the human genome. It consists of three algorithms: BWA-backtrack, BWA-SW, and BWA-MEM. The first algorithm is designed for Illumina sequence reads up to 100bp. BWA-MEM and BWA-SW share similar features such as long-read support and split alignment, but BWA-MEM (maximal exact matches), which is the latest, is generally recommended for high-quality queries as it is faster and more accurate (https://github.com/lh3/bwa) [8, 9]. The splice-unaware alignment algorithms are recommended for species like bacteria.

A detailed description of the usage of BWA can be found in the BWA manual (http://bio-bwa.sourceforge.net/bwa.shtml).

```
bwa mem /path/to/GenomeIndices/BWA/GRCh38/*.fa reads.fq [mates.fq]
```

```
Description of parameters:
*.fa                    path to BWA index file
reads.fq                fastq input files (unpaired or first mate if
                        paired-end)
mates.fq                fastq input files (second mate if paired-end)
```

9.2.3.6 HISAT2

HISAT (and its newer version HISAT2) is the next generation of spliced aligner from the same group that has developed TopHat. HISAT uses an indexing scheme based on the Burrows–Wheeler transform and the Ferragina–Manzini (Fm) index, employing two types of indices for alignment: a whole-genome Fm index to anchor each alignment and numerous local Fm indexes for very rapid extensions of these alignments (https://github.com/DaehwanKimLab/hisat) [15].

HISAT most interesting features include its high speed and its low memory requirement. HISAT is an open-source software freely available. A detailed description of the usage of HISAT can be found in the HISAT manual (https://ccb.jhu.edu/software/hisat2/manual.shtml).

```
hisat2 [options]* -x <hisat2-idx> {-1 <m1> -2 <m2> } | -U <r>
```

```
Description of parameters:
-x                      path to Hisat2 index
-U <r>                  unpaired reads
-1 <m1>                 fastq input files (first mate)
-2 <m2>                 fastq input files (second mate)
```

Take Home Message

- Sequence alignment is the process of comparing and detecting distances/similarities between biological sequences.
- Dynamic programming technique can be applied to global alignments by using methods such as global and local alignment algorithms.
- The value that measures the degree of sequence similarity is called the *alignment score*.

(continued)

- Sequence alignment includes calculating the so-called *edit distance,* which generally corresponds to the minimal number of substitutions, insertions, and deletions needed to turn one sequence into another.
- The choice of a sequencing read alignment tool depends on your goals and the specific case.

Review Questions

1. Assume you are performing a mapping with STAR. After the process, where do you find the information how many reads have mapped only at one position of the reference genome?
2. Which of the following does not describe local alignment?
   A. A local alignment aligns a substring of the query sequence to a substring of the target sequence.
   B. A local alignment is defined by maximizing the alignment score, so that deleting a column from either end would reduce the score, and adding further columns at either end would also reduce the score.
   C. Local alignments have terminal gap.
   D. The substrings to be examined may be all of one or both sequences; if all of both are included, then the local alignment is also global.
3. Which of the following does not describe BLAST?
   A. It stands for Basic Local Alignment Search Tool.
   B. It uses word matching like FASTA.
   C. It is one of the tools of the NCBI.
   D. Even if no words are similar, there is an alignment to be considered.
4. Which of the following does not describe dynamic programming?
   A. The approach compares every pair of characters in the two sequences and generates an alignment, which is the best or optimal.
   B. Global alignment algorithm is based on this method.
   C. Local alignment algorithm is based on this method.
   D. The method can be useful in aligning protein sequences to protein sequences only.
5. Which of the following is not a disadvantage of Needleman–Wunsch algorithm?
   A. This method is comparatively slow.
   B. There is a need of intensive memory.
   C. This cannot be applied on genome sized sequences.
   D. This method can be applied to even large sized sequences.
6. Alignment algorithms, both global and local, are fundamentally similar and only differ in the optimization strategy used in aligning similar residues.
   A. True.
   B. False.

7. The function of the scoring matrix is to conduct one-to-one comparisons between all components in two sequences and record the optimal alignment results.
   A. True.
   B. False.

Answers to Review Questions

1. *Log.final.out* in the output directory of alignment file (.bam); 2. C; 3. D; 4. D; 5. D; 6. A; 7. A

Acknowledgements We are grateful to Dr. Richa Bharti (Bioinformatician at TUM Campus Straubing, Germany) and Dr. Philipp Torkler (Senior Bioinformatics Scientist, *Exosome Diagnostics, a Bio-Techne brand*, Munich, Germany) for critically reading this text. We thank for correcting our mistakes and suggesting relevant improvements to the original manuscript.

References

1. Wilbur WJ, Lipman DJ. Rapid similarity searches of nucleic acid and protein data banks. Proc Natl Acad Sci U S A. 1983;80(3):726–30.
2. Pearson WR, Lipman DJ. Improved tools for biological sequence comparison. Proc Natl Acad Sci U S A. 1988;85(8):2444–8.
3. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ. Basic local alignment search tool. J Mol Biol. 1990;215(3):403–10.
4. Canzar S, Salzberg SL. Short read mapping: an algorithmic tour. Proc IEEE Inst Electr Electron Eng. 2017;105(3):436–58.
5. Fonseca NA, Rung J, Brazma A, Marioni JC. Tools for mapping high-throughput sequencing data. Bioinformatics. 2012;28(24):3169–77.
6. Needleman SB, Wunsch CD. A general method applicable to the search for similarities in the amino acid sequence of two proteins. J Mol Biol. 1970;48(3):443–53.
7. Smith TF, Waterman MS. Identification of common molecular subsequences. J Mol Biol. 1981;147(1):195–7.
8. Li H, Durbin R. Fast and accurate short read alignment with Burrows-Wheeler transform. Bioinformatics. 2009;25(14):1754–60.
9. Li H, Durbin R. Fast and accurate long-read alignment with Burrows-Wheeler transform. Bioinformatics. 2010;26(5):589–95.
10. Dobin A, Gingeras TR. Optimizing RNA-seq mapping with STAR. Methods Mol Biol. 2016;1415:245–62.
11. Langmead B. Aligning short sequencing reads with Bowtie. Curr Protoc Bioinformatics. 2010;32:11–7.
12. Langmead B, Trapnell C, Pop M, Salzberg SL. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biol. 2009;10(3):R25.
13. Langmead B, Salzberg SL. Fast gapped-read alignment with Bowtie 2. Nat Methods. 2012;9(4):357–9.
14. Kim D, Pertea G, Trapnell C, Pimentel H, Kelley R, Salzberg SL. TopHat2: accurate alignment of transcriptomes in the presence of insertions, deletions and gene fusions. Genome Biol. 2013;14(4):R36.
15. Kim D, Langmead B, Salzberg SL. HISAT: a fast spliced aligner with low memory requirements. Nat Methods. 2015;12(4):357–60.