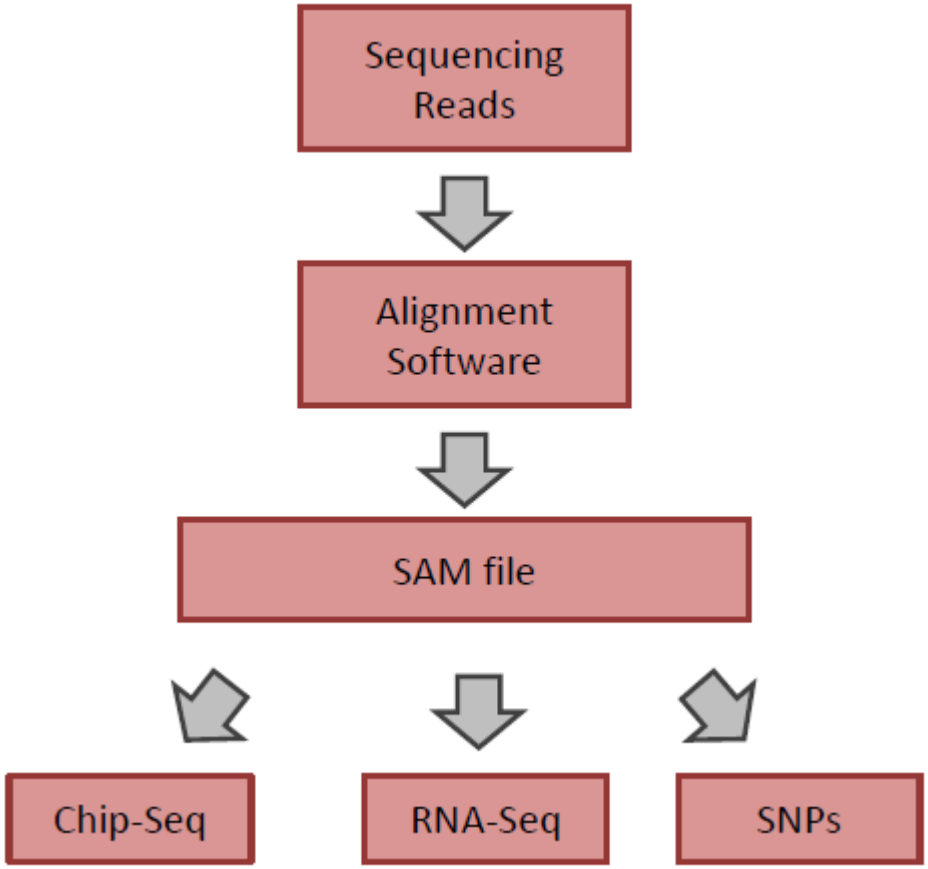
Working with NGS Data

10th August 2012

Overview


Sequencing Reads
Alignment Software
SAM file
Chip-Seq
RNA-Seq
SNPs

FASTQ

- Common file format for sharing sequencing read data

```
$ head SRR094861.fastq
@SRR094861.1 HWI-EAS206_23_30H0YAAXX:5:1:709:724 length=36
ACCTCCCGCCCCCNACNGCNCCCCCGTGCACCTCCC
+SRR094861.1 HWI-EAS206_23_30H0YAAXX:5:1:709:724 length=36
55555555555555!%5!55!555555$555(.$2.2
@SRR094861.2 HWI-EAS206_23_30H0YAAXX:5:1:1460:1251 length=36
AAACAAGCTAACATGACTAACACCCTTAATTCCATC
+SRR094861.2 HWI-EAS206_23_30H0YAAXX:5:1:1460:1251 length=36
5555555555555555555555555555121-11
@SRR094861.3 HWI-EAS206_23_30H0YAAXX:5:1:899:1936 length=36
AACAGTCTGATTAAAAAATGGGCCAAAGAGCTTAAC
```

1. sequence id
2. sequence content
3. + (placeholder)
4. quality string

- Simple but not well specified!

PHRED quality scores

- The reported quality indicates the probability of an error

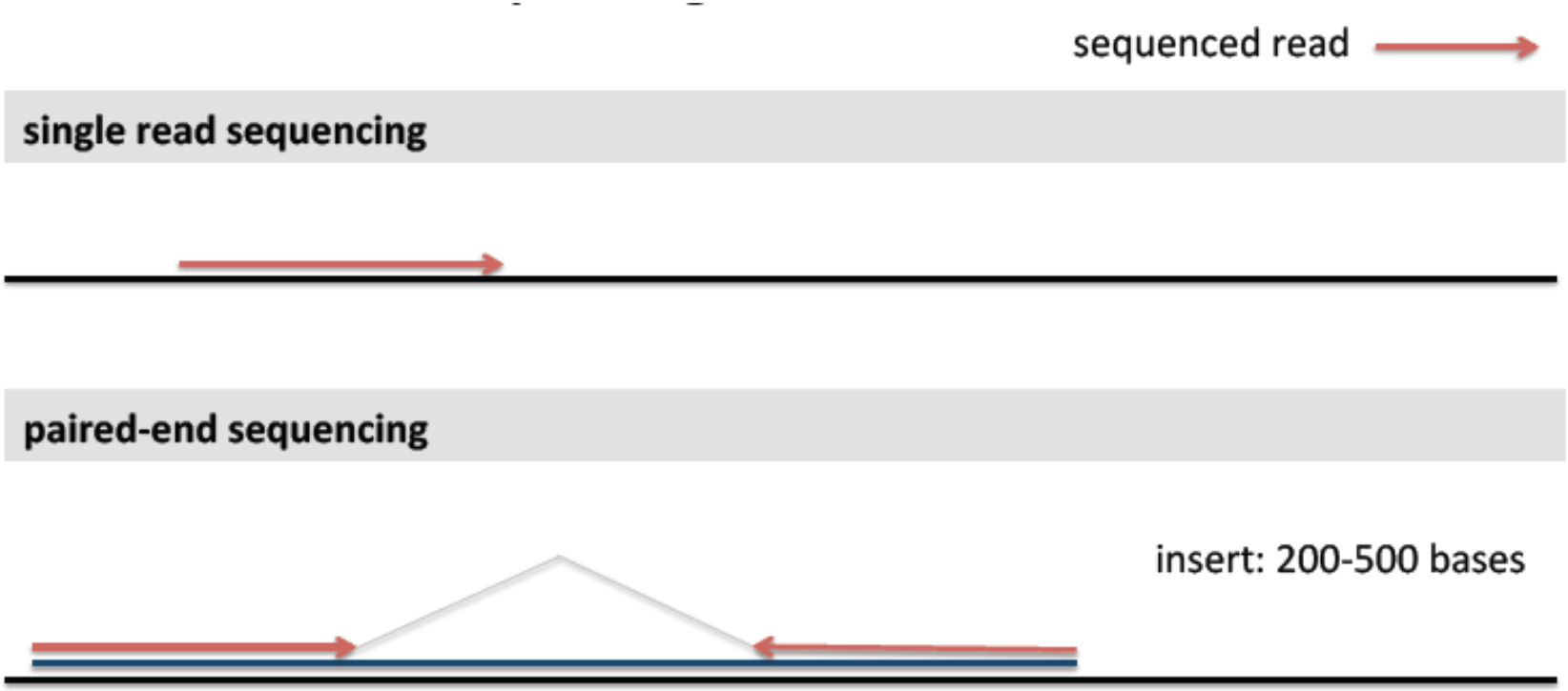
$$Q = -10 \times \log(P) \qquad \text{or} \qquad P = 10^{-Q/10}$$

$Q = 10 \rightarrow P = 10^{-1} = 1/10$ (one in ten)

$Q = 40 \rightarrow P = 10^{-4} = 1/10000$ (one in ten thousand)

- FASTQ: encoded as character (e.g. '5' = 53 - 33)
- Range not defined!

Paired end data



Paired end in FASTQ

- There is no standardized way so save paired end data in FASTQ files!!
- Two possibilities:
  - Use two files: reads_1.fq and reads_2.fq
  - Use one file
- Most programs don't check the reads names to find a matching pair
- Simple to convert. You just have to know what the program you are using expects.

Working with FATSQ files

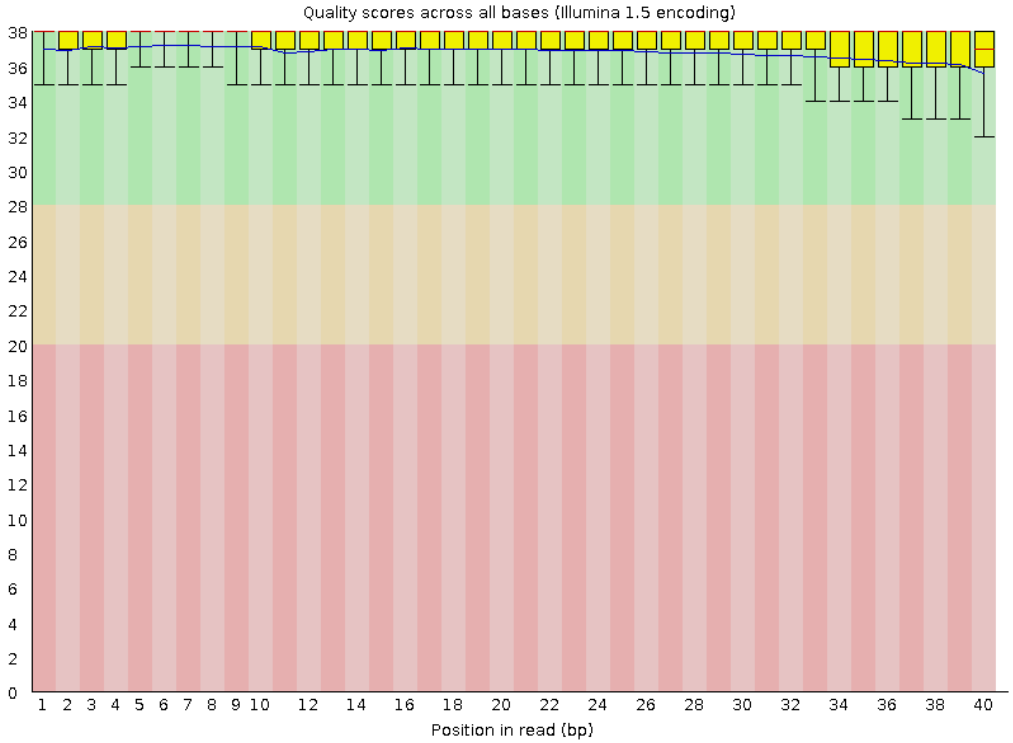
- Lots of tools:
  - fastx
  - fastqc
  - …
- Ask google!

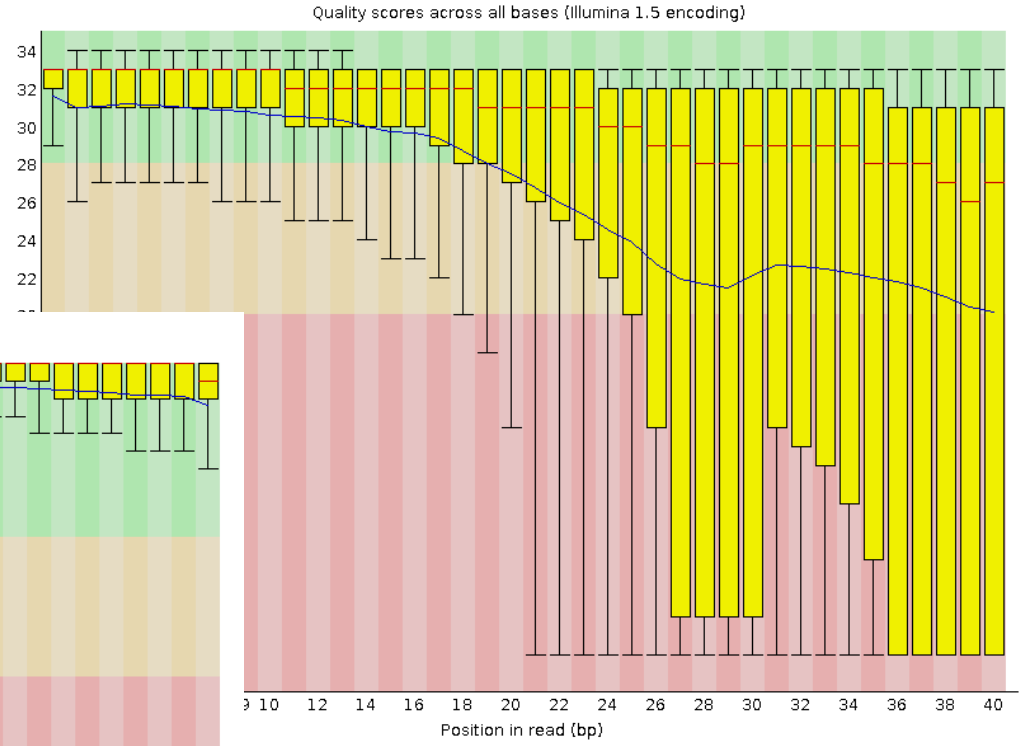
FASTQC

- A quality control tool for high throughput sequence data
- Written in Java: runs on Linux/Windows/Mac/…
- Easy to use
- GUI and command line mode

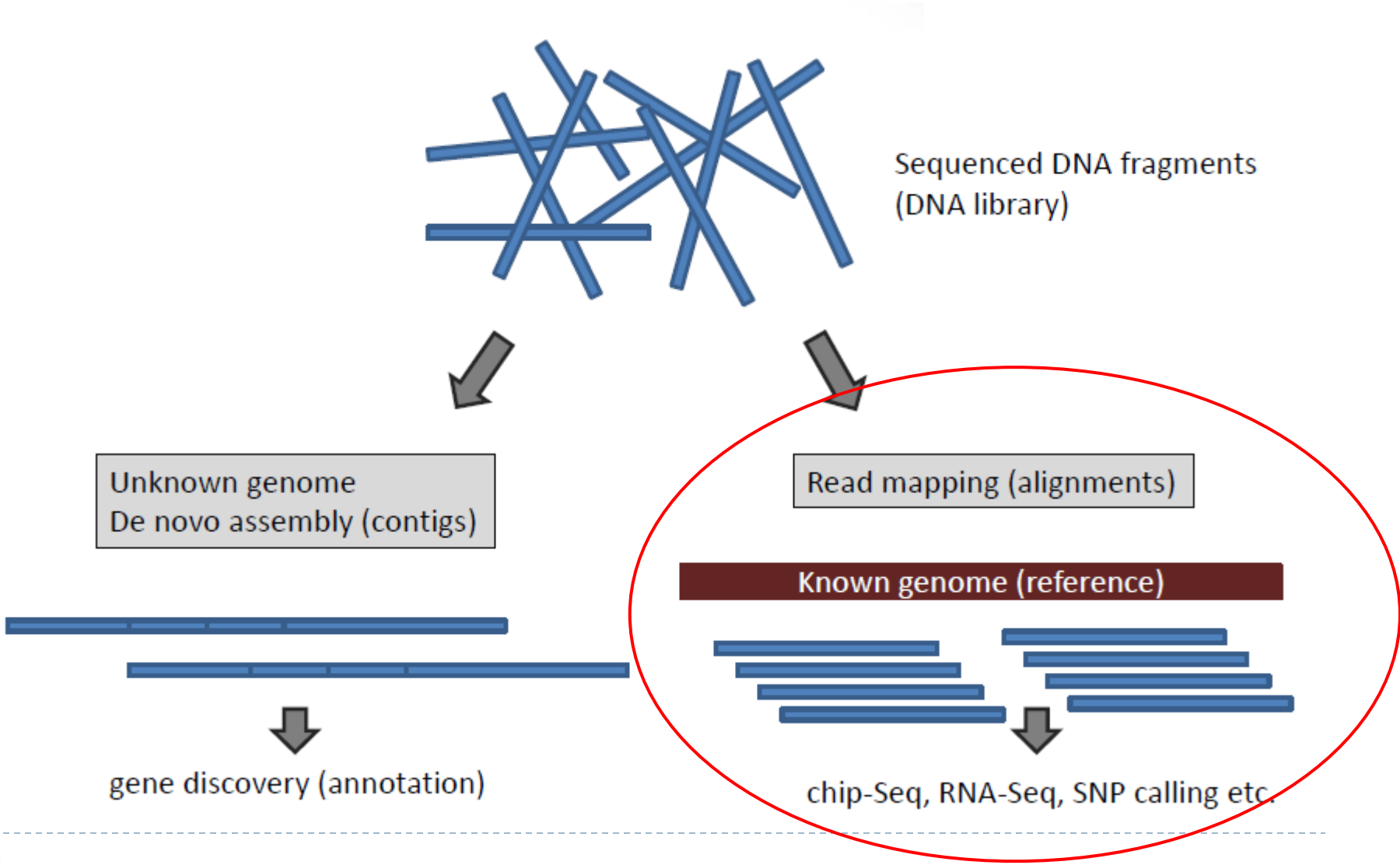
- http://www.bioinformatics.babraham.ac.uk/projects/fastqc/

Per base sequence quality





Processing sequencing data

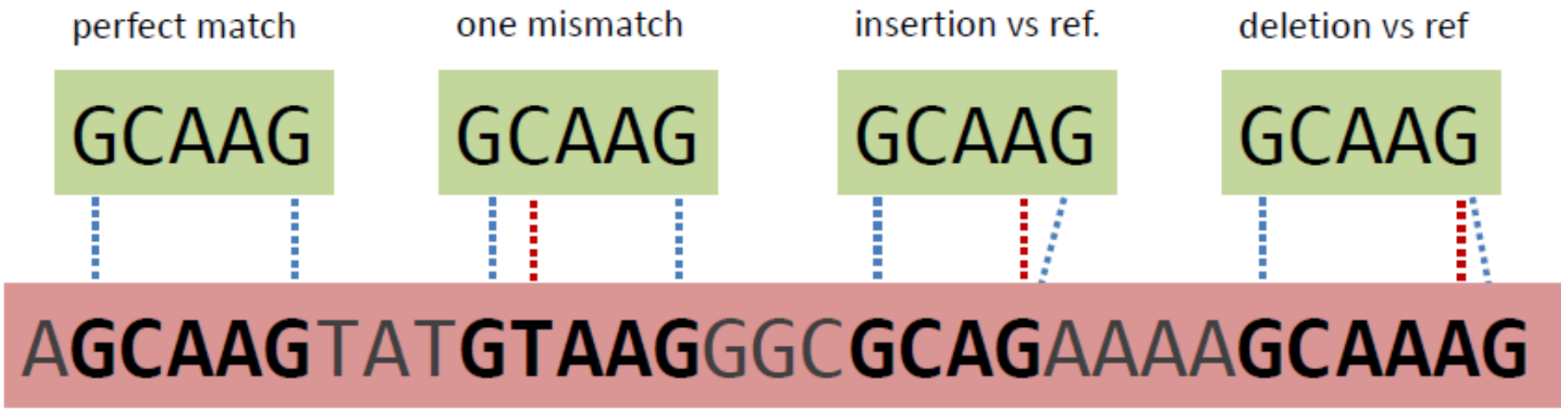


Sequence Alignments

- Arranging two or more sequences such as to maximize the length of the common regions between the two
- Well developed field of research

- High throughput sequencing poses special constraints: a very large number of very short reads – traditional methods were not feasible

Alignment concepts

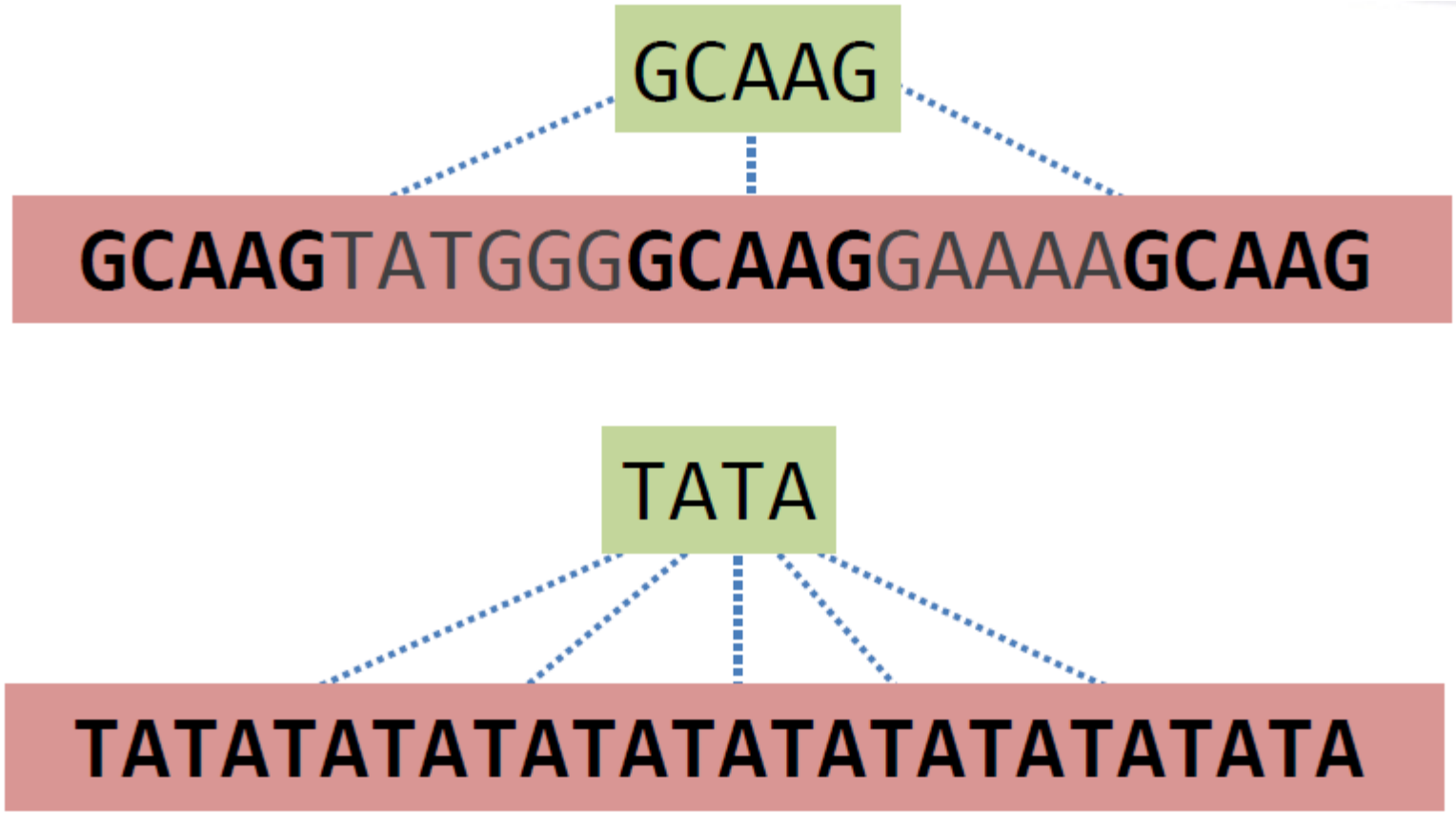


NOTE: mismatches or indels can be longer than 1 base!

It gets complicated very quickly

Alignment scoring depends on mismatch scoring (different across bases!), gap open, gap extension penalties

Repetitive and low complexity regions



Short Read Mappers (aligners)

- Using heuristics to quickly identify locations (hits) where the reads match
- Heuristics -> not all hits will be found
- Tradeoff:

  resource usage vs **speed** vs **accuracy** vs **usability**

- In addition: sequencing technologies develop extremely fast (e.g. read length)

A few popular aligners

- BWA, bowtie(2), SOAP2, Shrimp, BFAST and many others

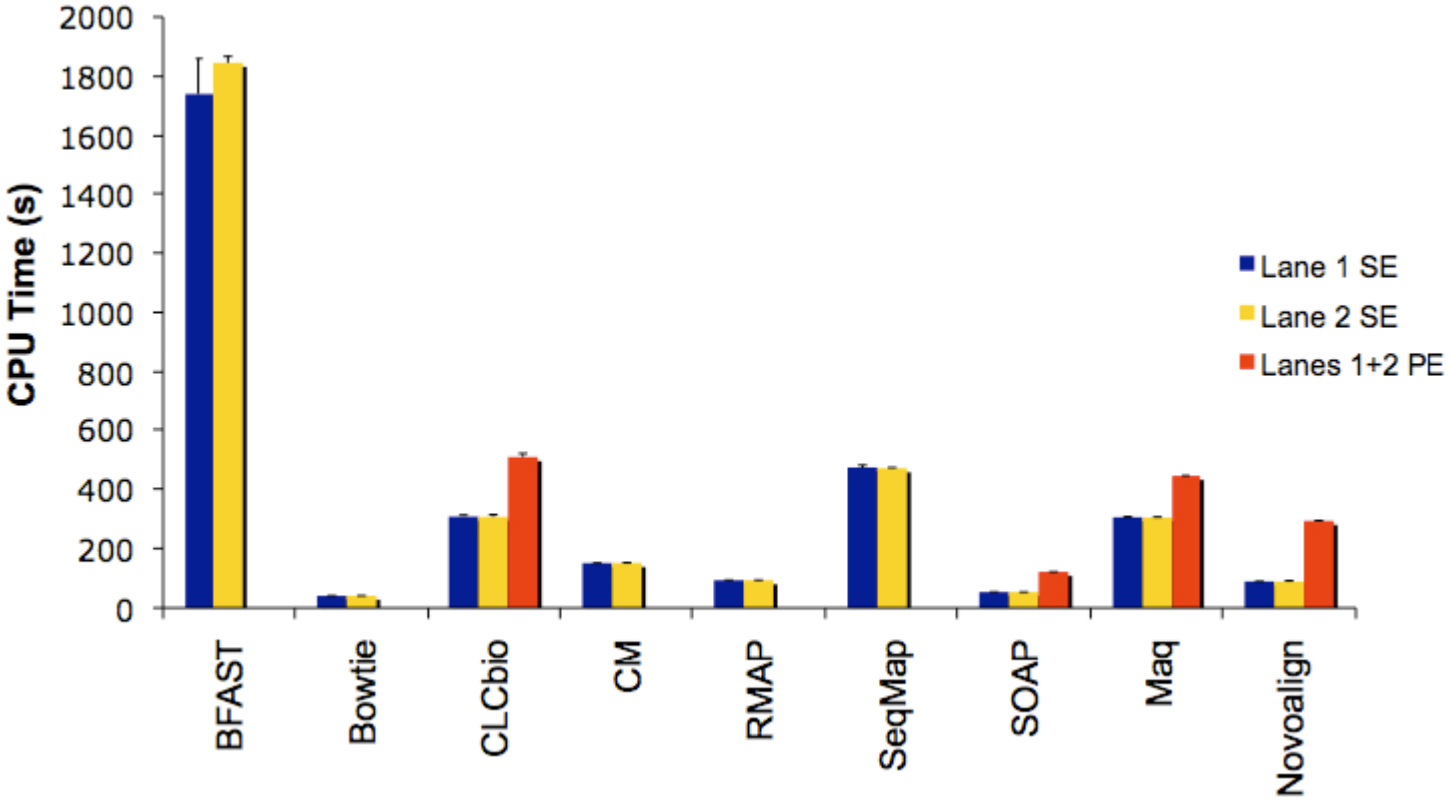
There is no single best tool, the issues to consider:

1. documentation → can we figure out how it works
2. input features → what type of input can it handle
3. reporting features → will it produce the type of output that we can use
4. performance → is it feasible to run on my resources

Aligner features

- Some cannot handle indels -> these tools are typically extremely fast and will run on a laptop for even large genomes (bowtie)
- May or may not use the quality score during the alignment
- Reporting alignment:
  - Unique alignments only
  - Best alignment above a cutoff
  - All possible alignment (slow)

CPU time to align 1 million reads

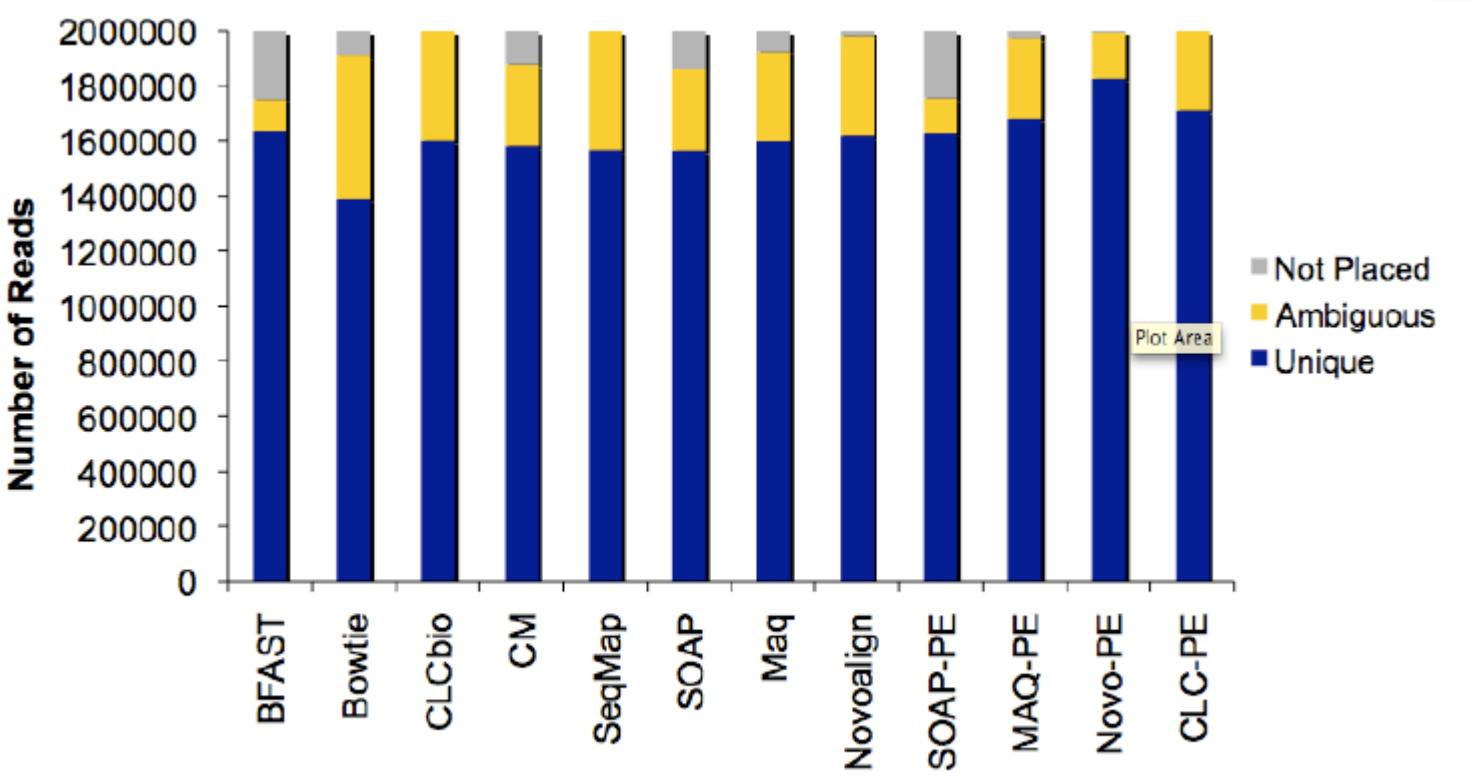


Example

- Computer: 4 cores
- Human genome

- 66 mio, 36bp (small dataset)
  - Shrimp2: ~30h
  - Stampy: 11h
  - BWA: 1h

Read placement for 2 million reads from the human genome



from: http://www.massgenomics.org/short-read-aligners

BWA

- Very popular
- Stable (well tested)

- Based on Burrows-Wheeler transformation
  - Fast
  - But: can handle only a small number of differences between the reads and the reference

- Well suited for genomes with a low number of polymorphisms (e.g. human)

Aligning reads with BWA

- Three step process (this is different for each aligner):

1. **Index the genome** – this only needs to be done once for each genome → **bwa index …**
2. **Create the alignment → bwa aln …**
3. **Report the alignment → bwa samse …**

Aligning reads with BWA

- Index
- Alignment
- Output (single end)

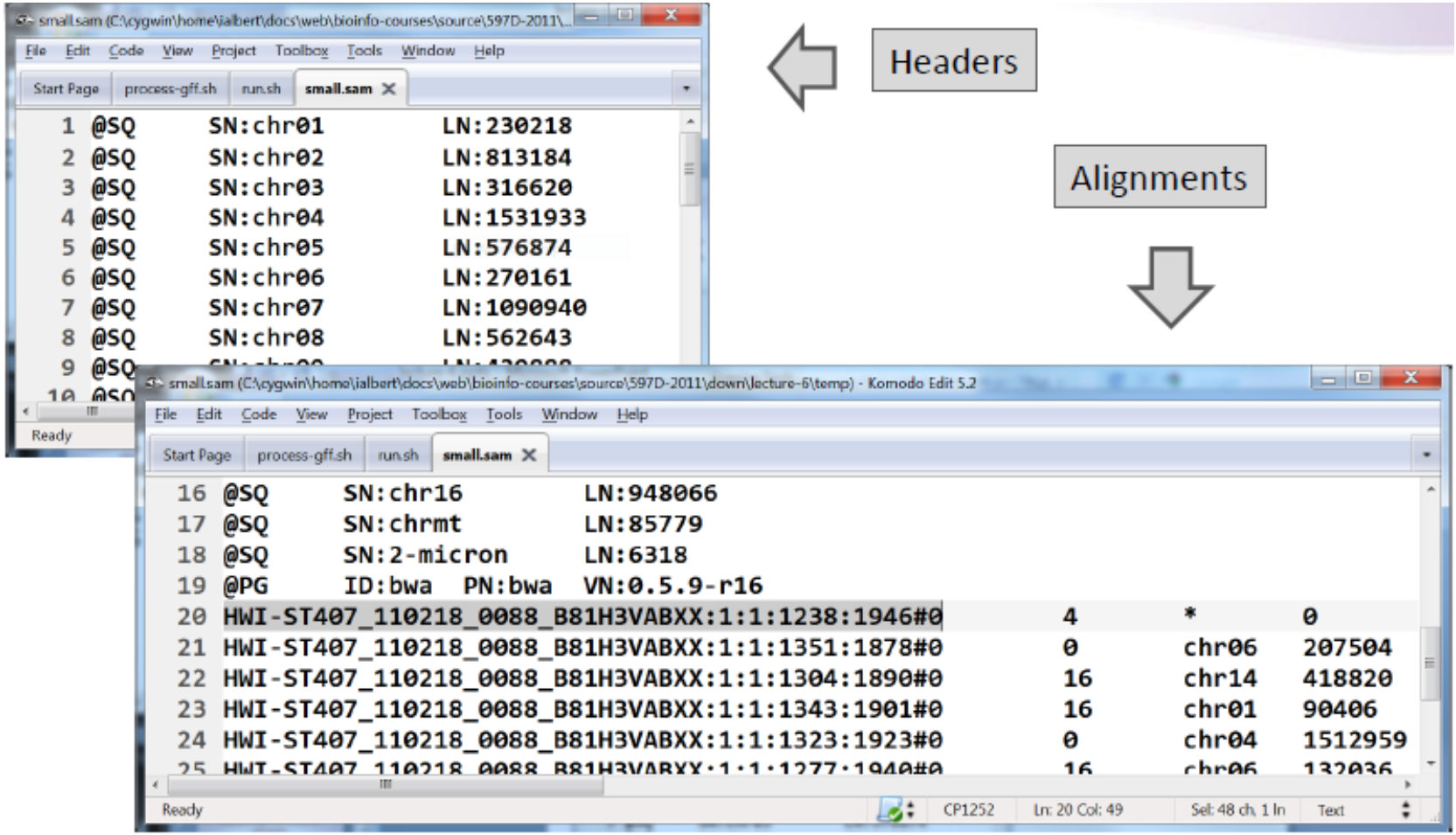
SAM: Sequence Alignment format

Published:
The Sequence Alignment/Map format and SAMtools by Heng Li et al
Bioinformatics 25, Volume 25, Issue 16, 2009

- SAM: A TAB-delimited text format consisting of a header section, which is optional, and an alignment section

- BAM: compressed version of SAM (same information)
  - Smaller but not human readable

The structure of a SAM file



The alignment columns

1.4 The alignment section: mandatory fields

Each alignment line has 11 mandatory fields. These fields always appear in the same order and must be present, but their values can be '0' or '*' (depending on the field) if the corresponding information is unavailable. The following table gives an overview of the mandatory fields in the SAM format:

Col	Field	Type	Regexp/Range	Brief description
1	QNAME	String	`[!-?A-~]{1,255}`	Query template NAME
2	FLAG	Int	$[0, 2^{16}-1]$	bitwise FLAG
3	RNAME	String	`\*|[!-()+-<>-~][!-~]*`	Reference sequence NAME
4	POS	Int	$[0, 2^{29}-1]$	1-based leftmost mapping POSition
5	MAPQ	Int	$[0, 2^{8}-1]$	MAPping Quality
6	CIGAR	String	`\*|([0-9]+[MIDNSHPX=])+`	CIGAR string
7	RNEXT	String	`\*|=|[!-()+-<>-~][!-~]*`	Ref. name of the mate/next segment
8	PNEXT	Int	$[0, 2^{29}-1]$	Position of the mate/next segment
9	TLEN	Int	$[-2^{29}+1, 2^{29}-1]$	observed Template LENgth
10	SEQ	String	`\*|[A-Za-z=.]+`	segment SEQuence
11	QUAL	String	`[!-~]+`	ASCII of Phred-scaled base QUALity+33

Example

(a)

```
coor    12345678901234  5678901234567890123456789012345
ref     AGCATGTTAGATAA**GATAGCTGTGCTAGTAGGCAGTCAGCGCCAT

r001+         TTAGATAAAGGATA*CTG
r002+       aaaAGATAA*GGATA
r003+     gcctaAGCTAA
r004+                   ATAGCT..............TCAGC
r003-                         ttagctTAGGC
r001-                                      CAGCGCCAT
```

(b)

```
@SQ SN:ref LN:45
r001 163 ref  7 30 8M2I4M1D3M = 37  39 TTAGATAAAGGATACTA *
r002   0 ref  9 30 3S6M1P1I4M *  0   0 AAAAGATAAGGATA    *
r003   0 ref  9 30 5H6M       *  0   0 AGCTAA   *   NM:i:1
r004   0 ref 16 30 6M14N5M    *  0   0 ATAGCTTCAGC       *
r003  16 ref 29 30 6H5M       *  0   0 TAGGC    *   NM:i:0
r001  83 ref 37 30 9M         =  7 -39 CAGCGCCAT         *
```

Working with SAM files

- Samtools
- Commands: view, index, sort, merge, flagstat, fixmate, …
- Example: View
  - Convert between SAM and BAM
  - Filter SAM and BAM files

- Another popular tool: picard tools
  - Written in java, sometimes faster (e.g. sort SAM file)
  - http://picard.sourceforge.net/command-line-overview.shtml

Samtools: Examples

- How many reads where aligned:

```
$ samtools view -S -F 0x4 SRR094861.sam | perl -F'\t' -ane 'print
if(/NM\:i\:0/)' | wc -l
17727982
```

- How many reads mapped to only one position:

```
$ samtools view -S -F 0x4 SRR094861.sam | perl -F'\t' -ane 'print
if(/NH\:i\:1[\t\n]/)' | wc -l
7736878
```

- Convert SAM to BAM:

```
$ samtools view -bS SRR094861.sam > SRR094861.bam
```

```
$ samtools sort SRR094861.bam SRR094861.sorted

$ samtools index SRR094861.sorted.bam
```

Summary

- SAM format:
  - http://samtools.sourceforge.net/SAM1.pdf
- Most important tools:
  - mapper: bwa, bowtie(2), stampy
  - samtools
  - picard tools
  - fastx
  - fastqc
  - IGV (genome browser written in java)

Resources

- Most slides were taken from:
  - http://evop.bioinf.uni-leipzig.de/?page_id=7
  - http://www.personal.psu.edu/iua1/courses/2011/2011-analyzing-high-througphut-sequencing-data
- Others:
  - http://www.bits.vib.be/index.php/training/124-linux-for-bioinformatics

BIOINFORMATICS: Short Read Archive

Task:

- Find and download the run **SRR094861** of the experiment with the accession SRX038963 from the **Short Read Archive** (SRA) and safe it in your folder.

BIOINFORMATICS: Short Read Archive

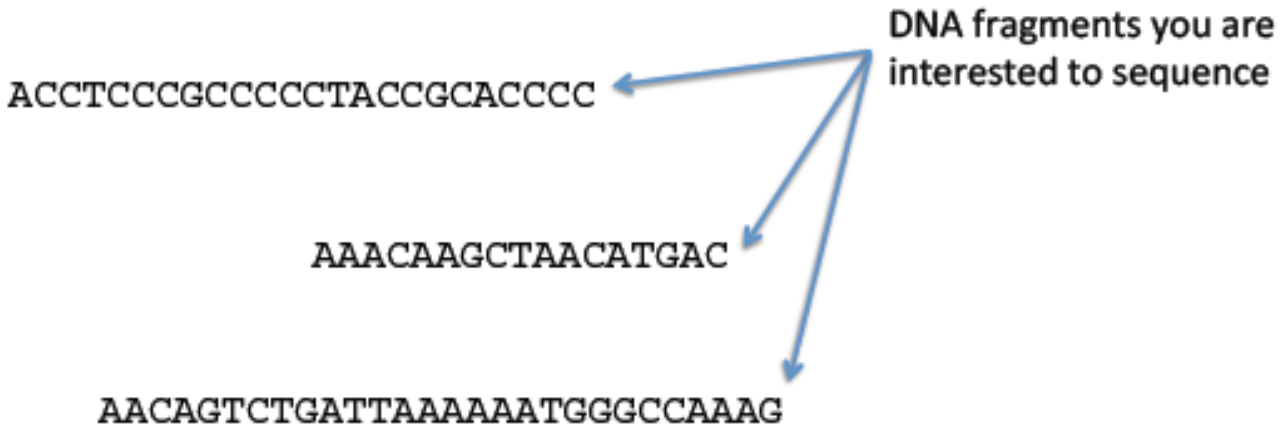
Task:

- Convert SRA format data into FASTQ using the **SRA toolkit** (http://www.ncbi.nlm.nih.gov/books/NBK47540/)

```
$ fastq-dump SRR094861.sra
```

BIOINFORMATICS: Adapter Clipping

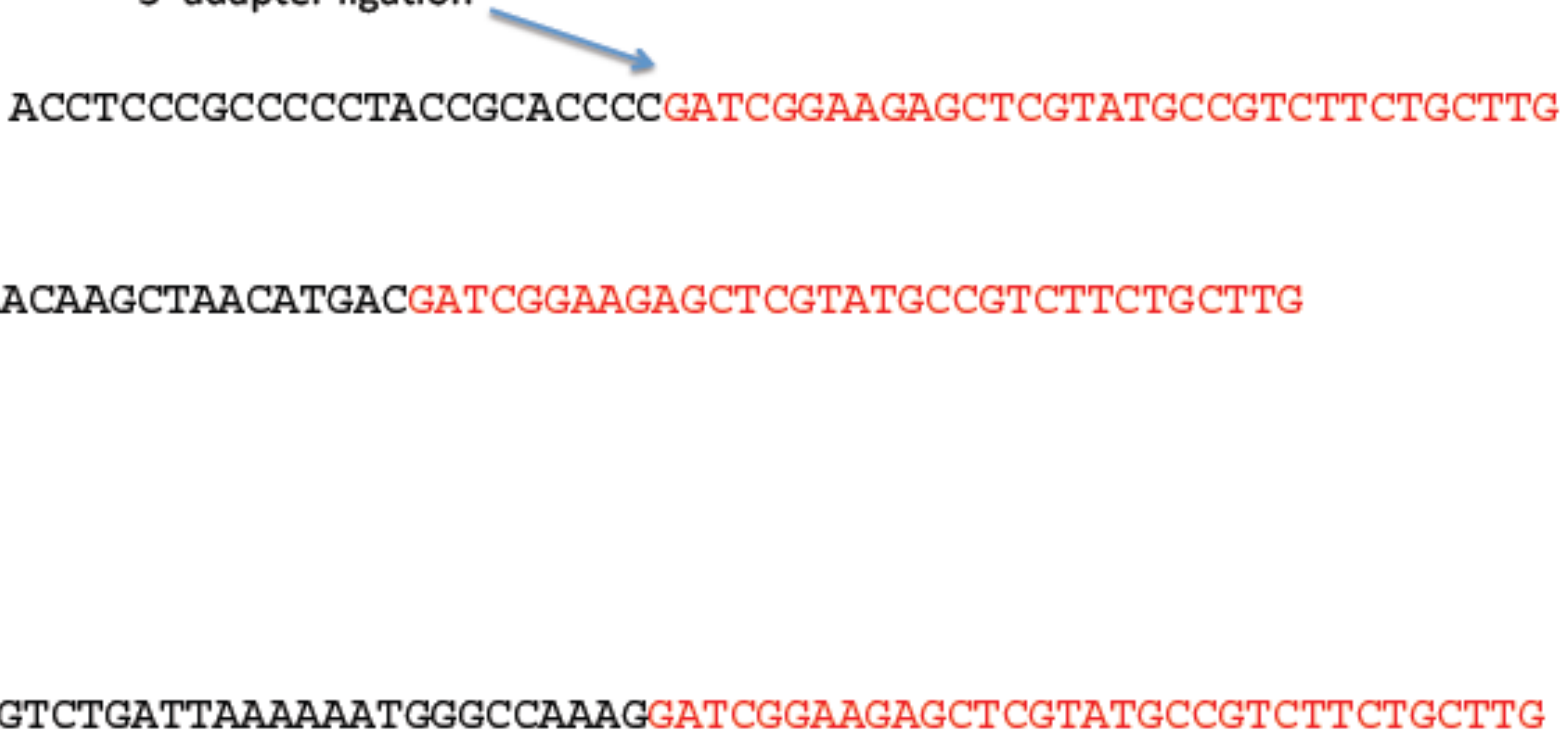
Library preparation:



BIOINFORMATICS: Adapter Clipping

Library preparation
(also a 5' adapter is ligated, but due to simplicity, it's not shown here; it will not be sequenced anyway):

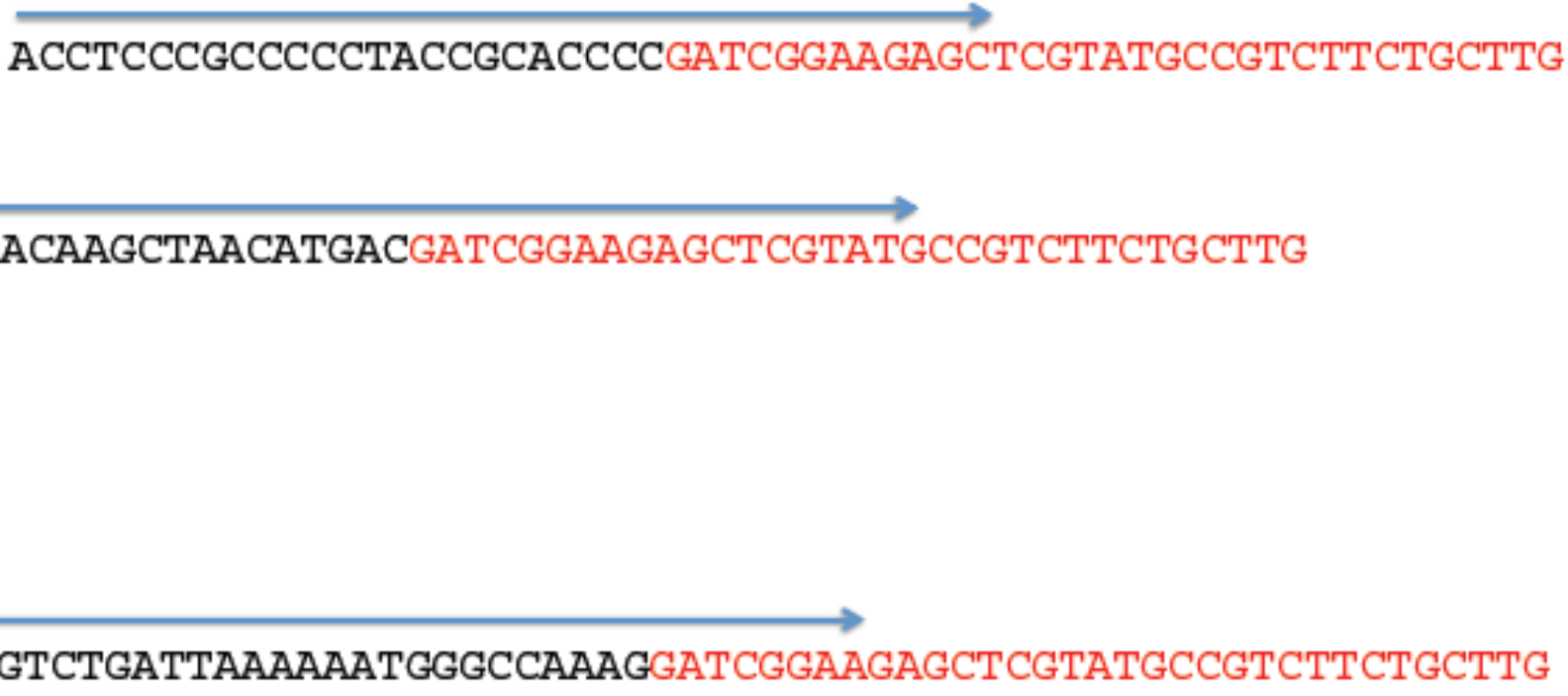
3' adapter ligation



BIOINFORMATICS: Adapter Clipping

Sequencing:

exact 36 nucleotides are sequenced (defined by your machine; here Illumina)



BIOINFORMATICS: Adapter Clipping

Data you get from the sequencing machine:

adapter:
GATCGGAAGAGCTCGTATGCCGTCTTCTGCTTG

reads:
ACCTCCCGCCCCCTACCGCNCCCCGATCGGAAGAGC

AAACAAGCTAACATGACGATCGGAAGAGCTCGTATG

AACAGTCTGATTAAAAAATGGGCCAAAGGATCGGAA

BIOINFORMATICS: Adapter Clipping

Align adapter to reads:

```
                adapter: GATCGGAAGAGCTCGTATGCCGTCTTCTGCTTG
ACCTCCCGCCCCCTACCGCNCCCCGATCGGAAGAGC

        adapter: GATCGGAAGAGCTCGTATGCCGTCTTCTGCTTG
AAACAAGCTAACATGACGATCGGAAGAGCTCGTATG

                adapter: GATCGGAAGAGCTCGTATGCCGTCTTCTGCTTG
AACAGTCTGATTAAAAAATGGGCCAAAGGATCGGAA
```

BIOINFORMATICS: Adapter Clipping

Clip adapter sequence:

ACCTCCCGCCCCCTACCGCNCCCC

AAACAAGCTAACATGAC

AACAGTCTGATTAAAAAATGGGCCAAAG

BIOINFORMATICS: Adapter Clipping

Task:

- Find out what the sequence of the used adapter is?

Problem:

- If the adapter sequence is not mentioned in the study summary, you can try to
  - find it on the manufacturers' website (here Illumina).
  - google it.
  - predict it yourself (e.g. with segemehl using the –Y parameter).

Sequences for Illumina Library Preparation:

Adapter
5' P-GATCGGAAGAGCTCGTATGCCGTCTTCTGCTTG

BIOINFORMATICS: Adapter Clipping

Task:

- Clip the adapter sequence using the fastx-toolkit (http://hannonlab.cshl.edu/fastx_toolkit/)

```
$ fastx_clipper -a GATCGGAAGAGCTCGTATGCCGTCTTCTGCTTG -Q 33 -l 15
-n –i SRR094861.fastq -o SRR094861.clipped.fastq
```

-a	ADAPTER string
-Q	quality scores offset
-l	discard sequences shorter than N nucleotides
-n	keep sequences with unknown (N) nucleotides
-o	FASTA/Q output file

SAM: Bitwise flags

2. FLAG: bitwise FLAG. Each bit is explained in the following table:

Bit	Description
0x1	template having multiple segments in sequencing
0x2	each segment properly aligned according to the aligner
0x4	segment unmapped
0x8	next segment in the template unmapped
0x10	SEQ being reverse complemented
0x20	SEQ of the next segment in the template being reversed
0x40	the first segment in the template
0x80	the last segment in the template
0x100	secondary alignment
0x200	not passing quality controls
0x400	PCR or optical duplicate

- Bit 0x4 is the only reliable place to tell whether the segment is unmapped. If 0x4 is set, no assumptions can be made about RNAME, POS, CIGAR, MAPQ, bits 0x2, 0x10 and 0x100 and the bit 0x20 of the next segment in the template.

SAM: Bitwise representation

1 = 00000001 → paired end read
2 = 00000010 → mapped as proper pair
4 = 00000100 → unmappable read
8 = 00001000 → read mate unmapped
16 = 00010000 → read mapped on reverse strand

The flag 11 → 1 + 2 + 8 = 0001011 (conditions 1, 2 and 8 satisfied)

It is used to save space – but it does make things a bit more difficult.
Usually very few flags are needed in practice – 0, 4, 16 are the most generic ones

If you need to construct a more complex flag search for explain SAM flags:

http://picard.sourceforge.net/explain-flags.html

Mapping Quality definition

- Phred score, identical to the quality measure in the fastq file. quality **Q**, probability **P**:

$$P = 10 \wedge (-Q / 10.0)$$

If **Q=30**, **P=1/1000** → on average, one of out 1000 alignments will be wrong

As good as this sounds it is not easy to compute such a quality.

Details of the mapping quality computation

- The repeat structure of the reference. Reads falling in repetitive regions usually get very low mapping quality.
- The base quality of the read. Low quality means the observed read sequence is possibly wrong, and wrong sequence may lead to a wrong alignment.
- The sensitivity of the alignment algorithm. The true hit is more likely to be missed by an algorithm with low sensitivity, which also causes mapping errors.
- Paired end or not. Reads mapped in pairs are more likely to be correct.

(*from the MAQ manual*)

BWA specific high scores

A read alignment with a mapping quality 30 or above usually implies

- The overall base quality of the read is good.
- The best alignment has few mismatches.
- The read has few or just one `good' hit on the reference, which means the current alignment is still the best even if one or two bases are actually mutations or sequencing errors.

BWA specific low scores

Surprisingly difficult to track down the exact behavior

- Q=0 → if a read can be aligned equally well to multiple positions, BWA will randomly pick one position and give it a mapping quality zero.

- Q=25 → the edit distance equals mismatches and is greater than zero

Visualizing the results

- IGV
- https://www.broadinstitute.org/igv/