

Short read alignment

Introduction to short read alignment, mapping with bowtie, bowtie2, and BWA, visualizing alignments

Required software

Install **bowtie** aligner

```
$ conda install -c bioconda bowtie
```

Install **samtools**

```
$ conda install -c bioconda samtools
```

Install seqkit

```
$ conda install -c bioconda seqkit
```

Download sra-toolkit binaries, Bioconda version is too old

Sra-toolkit binaries are located at <https://github.com/ncbi/sra-tools/wiki/02.-Installing-SRA-Toolkit>

```
$ mkdir ~/programs
```

```
$ cd ~/programs
```

```
$ wget https://ftp-trace.ncbi.nlm.nih.gov/sra/sdk/current/sratoolkit.current-ubuntu64.tar.gz
```

```
$ tar -xzf sratoolkit.current-ubuntu64.tar.gz
```

```
$ rm sratoolkit.current-ubuntu64.tar.gz
```

```
$ ls -lh ~/programs/sratoolkit.3.0.1-ubuntu64/bin # take a look at the binaries
```

Check if the binaries work

```
$ ~/programs/sratoolkit.3.0.1-ubuntu64/bin/prefetch -h
```

Required software

Install ***bowtie2*** aligner

```
$ conda install -c bioconda bowtie2
```

Install BWA aligner

```
$ conda install -c bioconda bwa
```

Install Integrated Genomic Viewer (IGV)

```
$ conda install -c bioconda igv
```

Introduction to short read alignments

What are the properties of short read alignments?

- The query sequences are short in range of 50 – 300 bp
- The reference sequences are much longer than the query, i.e. a set of short sequencing reads are matched to a genome or to a set of transcripts
- Normally there is a large number of short sequence queries (millions) originating from high throughput sequencing platforms
- Short read aligners solve the problem of locating the best match to reference as fast as possible, as opposed to trying every possible alternative to find an optimal alignment
- Short reads aligners are not guaranteed to determine a best match or to locate all possible matches

Introduction to short read alignments

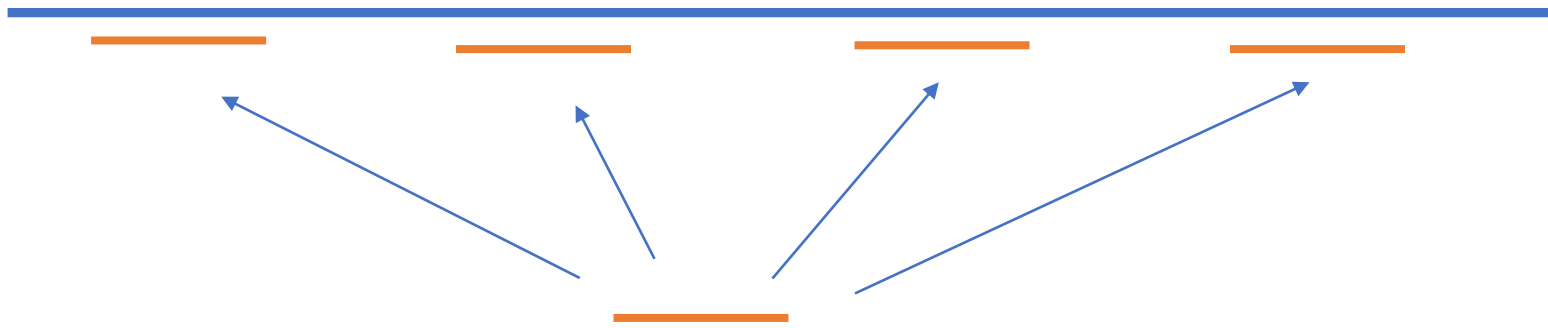
Short read alignment software

- The tools used for this task are called short read aligners or short read mappers
- Short read aligners are much faster than BLAST, and can match thousands of sequences per second against Human genome
- They're behavior can be tuned with numerous options
- Different aligners may produce different results
- NGS analysis may include looking for exact matches, partial matches, multiple alignments, detecting gaps and mismatches
- Different aligners and different option combinations of the same aligner allow to gain efficiency in some NGS analysis domains while losing them in others

Introduction to short read alignments

Software limitations to keep in mind

- The query sequences must be quite similar to a target, short read algorithm typically quit searching after a certain similarity threshold
- They will not work with the query sequences that are too short or too long, the size limitation should be described in the software manual
- Repetitive regions cause problems for short read aligners, for example if the reference contains lots of AAAAAAAAAAAAAA repeats and a short sequence is AAAAAA, we would not be able to tell where this read had originated from



Repeat problem –
we cannot really
tell where the
repetitive
sequence is
located

Introduction to short read alignments

A difference between mapping and alignment

- Mapping solves a problem of locating the genomic origin of the read
- The alignment tries to find the optimal arrangement of bases between sequences
- Mapping oriented methods forgo search for optimal arrangement for efficient read placement
- ChIP-seq, RNA-seq, ATAC-seq rely on mapping, while variant calling also requires alignment

Introduction to short read alignments


What alignment tools (short read and others) are available

Take a look at a comprehensive table of 107 alignment tools published since 1988 to 2020

<https://genomebiology.biomedcentral.com/articles/10.1186/s13059-021-02443-7/tables/1>

Review | [Open Access](#) | [Published: 26 August 2021](#)

Technology dictates algorithms: recent developments in read alignment

[Mohammed Alser](#), [Jeremy Rotman](#), [Dhriti Deshpande](#), [Kodi Taraszka](#), [Huwenbo Shi](#), [Pelin Icer Baykal](#),
[Harry Taegyun Yang](#), [Victor Xue](#), [Sergey Knyazev](#), [Benjamin D. Singer](#), [Brunilda Balliu](#), [David Koslicki](#), [Pavel Skums](#),
[Alex Zelikovsky](#), [Can Alkan](#), [Onur Mutlu](#) & [Serghei Mangul](#) 

[Genome Biology](#) **22**, Article number: 249 (2021) | [Cite this article](#)

10k Accesses | **16** Citations | **76** Altmetric | [Metrics](#)

Introduction to short read alignments

Popular and established short read aligners

Name	Link	Domain	Comment	Cited
HISAT2	http://daehwankimlab.github.io/hisat2/	RNA-seq	Splice-aware aligner popular, my primary choice for RNA-seq	4212
STAR	https://github.com/alexdobin/STAR	RNA-seq	Splice-aware aligner, used by both researchers and genome centers	29749
BWA	https://github.com/lh3/bwa	DNA-seq	Probably most popular in variant calling and other domains of DNA-seq (WGS, exome seq, ChIP-seq, etc)	38636
bowtie2	https://github.com/BenLangmead/bowtie2	DNA-seq	Another good and well cited DNA-seq alignment	38517
minimap2	https://github.com/lh3/minimap2	DNA-seq	Recommended for long reads from PacBio and Oxford nanopore	5475
BBMap	https://sourceforge.net/projects/bbmap/	DNA- and RNA-seq	Versatile aligner written in Java	1037

Introduction to short read alignments

Picking the right aligner

- Can the aligner handle your input data (paired- or single-end, long vs short, sequencing platform, etc.)?
- What is the output, for example, does it produce the alignments in SAM format?
- How does it handle gaps?
- Will it work on your system?
- Is it well cited, supported by developers and mature?
- Is it recommended by the developers and academic community for your use case?
- Is it recommended in “best practice” guides (for example, Best Practice Workflows GATK)?
- Additional information provided: splicing, soft-clipping, chimeric alignments, etc?

Mapping with Bowtie

Practice short read alignments with **bowtie**

- Disclaimer: we use bowtie as introductory software for training, use **bowtie2** or **BWA** for actual genomic analysis!
- Bowtie is a fast un-gapped (does not detect gaps) aligner that was popular once, but now is superseded by **bowtie2** that allows gaps

Software | [Open Access](#) | [Published: 04 March 2009](#)

Ultrafast and memory-efficient alignment of short DNA sequences to the human genome

[Ben Langmead](#) , [Cole Trapnell](#), [Mihai Pop](#) & [Steven L Salzberg](#)

[Genome Biology](#) **10**, Article number: R25 (2009) | [Cite this article](#)

285k Accesses | **14602** Citations | **98** Altmetric | [Metrics](#)

Link to **bowtie** manual

<https://bowtie-bio.sourceforge.net/manual.shtml>

Let's open and read bowtie manual, it's well written and concise!

Mapping with Bowtie

Short read alignment process

- Typically, short read alignment includes 2 steps
- Creating a reference index
- Mapping the reads to the reference

Let's download a reference and create an index

We will use E. coli reference genome shown the screenshot:

Escherichia coli str. K-12 substr. MG1655, complete genome

NCBI Reference Sequence: NC_000913.3

[FASTA](#) [Graphics](#)

[Go to:](#) ☐

LOCUS	NC_000913	4641652 bp	DNA	circular	CON 09-MAR-2022
DEFINITION	Escherichia coli str. K-12 substr. MG1655, complete genome.				
ACCESSION	NC_000913				
VERSION	NC_000913.3				

Mapping with Bowtie

Short read alignment process

Download and index genome

Make sure you are in **sandbox/** directory

```
$ mkdir -p GENOMES/E_coli # -p option will allow to create subdirectories
```

```
$ ls -lh GENOMES
```

```
$ cd GENOMES/E_coli
```

```
$ esearch -db nucleotide -query "NC_000913.3" | efetch -format fasta > Ecoli_K12.fasta
```

```
$ ls -lh # Check the result
```

Create E.coli genome index

Link to indexing section of the bowtie manual:

<https://bowtie-bio.sourceforge.net/manual.shtml#the-bowtie-build-indexer>

Bowtie index command syntax:

```
bowtie-build [options] <reference_in> <ebwt_base>
```

Mapping with Bowtie

Short read alignment process

Build bowtie index

```
$ mkdir bowtie_index # Create index directory
```

```
$ cd bowtie_index
```

```
$ bowtie-build ../Ecoli_K12.fasta Ecoli_K12
```

```
$ ls -lh # Check the files created
```

```
$ cd ../../
```

```
$ echo "GENOMES directory content" > README_GENOMES.txt # Let's create documentation for our  
GENOMES directory
```

```
$ echo "E_coli/ : Escherichia coli str. K-12 substr. MG1655, complete genome. NCBI Reference  
Sequence: NC_000913.3" >> README_GENOMES.txt
```

```
$ cat README_GENOMES.txt # view resulting document
```

```
$ cd ../ # Return to sandbox/ directory
```

Mapping with Bowtie

Short read alignment process

Mapping the reads

First, let's take a look at some of ***bowtie mapping*** options

Options:

-v, -n, -e, -l : this options decide which alignments are considered valid

-k, -a, -m, -M, --best, --strata : decide which alignments will be reported

-q, -f, -c, --phred64-quals : some input options

--al, --un, --suppress, -t : some output options

-S, --no-unal, --mapq : some SAM options

-p : performance options, parallelization

Mapping with Bowtie

Short read alignment process

Most useful options (in my opinion):

- **-v <INT>** – number of mismatches allowed in the entire read
- **-n <INT>** – number of mismatches allowed in *l* left-most reads set by **-l** option
- **-l <INT>** – seed length of *l* (number of left-most bases to be matched exactly)
- **-v** and **-n** modes are mutually exclusive
- **-k <INT>** - report up to *<INT>* valid alignments
- **-m <INT>** – suppress all reads with number of hits exceeding a number specified by **-m** option
- **--best** – guarantees to find best alignment at the expense of speed (I recommend to always use this option)
- **--tryhard** – try hard to find valid alignments, if they exist (very slow!)

Mapping with Bowtie

Short read alignment process

Most useful options (in my opinion):

- **-v <INT>** – number of mismatches allowed in the entire read
- **-n <INT>** – number of mismatches allowed in *l* left-most reads set by **-l** option
- **-l <INT>** – seed length of *l* (number of left-most bases to be matched exactly)
- **-v** and **-n** modes are mutually exclusive
- **-k <INT>** - report up to *<INT>* valid alignments
- **-m <INT>** – suppress all reads with number of hits exceeding a number specified by **-m** option
- **--best** – guarantees to find best alignment at the expense of speed (I recommend to always use this option)
- **--tryhard** – try hard to find valid alignments, if they exist (very slow!)

Mapping with Bowtie

Short read alignment process

- Let's practice bowtie alignments with some examples.
- The examples are from bowtie manual with some modifications

Map a single sequence to E_coli genome with 2 mismatches allowed across full length of the sequence
`$ bowtie -a -v 2 GENOMES/E_coli/bowtie_index/Ecoli_K12 -c ATGCATCATGCGCCAT # -a – output all valid alignments, -v 2 – allow up to 2 mismatches across a full length of the sequence, -c – get the sequence from the command line`

Output format:

Col 1. Query id

Col 2. Reference strand the read was mapped to – sense/antisense

Col 3. Reference sequence id

Col 4. 0-based start of the match

Col 5. Query sequence (reverse complemented if reference is '-')

Col 6. Read qualities

Col 7. Contains prescribed ceiling if –M option was set and the ceiling was exceeded

Col 8. Comma separated mismatch descriptors

Mapping with Bowtie

Short read alignment process

More examples:

We can change the format of default output with **--suppress** option

```
$ bowtie -a -v 2 GENOMES/E_coli/bowtie_index/Ecoli_K12 --suppress 1,2,3,6,7 -c ATGCATCATGCGCCA #
```

We suppressed the columns that do not carry any additional information when mapping a single sequence

Let's limit the number of valid alignments reported to 3

```
$ bowtie -k 3 -v 2 GENOMES/E_coli/bowtie_index/Ecoli_K12 --suppress 1,2,3,6,7 -c ATGCATCATGCGCCA
```

Repeat this command a couple of times, do the reported alignments change?

Combine **-k**, **--best** and **--strata** to report only a specified number of best alignments within a “stratum”

```
$ bowtie -k 3 -v 2 --best --strata GENOMES/E_coli/bowtie_index/Ecoli_K12 --suppress 1,2,3,6,7 -c ATGCATCATGCGCCA #
```

in this case, the “stratum” is defined by the number of mismatches, notice that only the reads in the best stratum (1 mismatch) are reported

Mapping with Bowtie

Short read alignment process

More examples:

Do not report the alignments for the reads with the number of valid matches that exceed the limit specified in **-m**

```
$ bowtie -a -m 20 -v 2 GENOMES/E_coli/bowtie_index/Ecoli_K12 --suppress 1,2,3,6,7 -c  
ATGCATCATGCGCCA # this read has 21 valid alignment with up to 2 mismatches
```

```
$ bowtie -a -m 22 -v 2 GENOMES/E_coli/bowtie_index/Ecoli_K12 --suppress 1,2,3,6,7 -c  
ATGCATCATGCGCCA # This will report all 21 valid alignments since it is below the -m limit of 22.
```

Typically, this option is used to exclude the reads mapping to repeats, for example **-m 1** will result in reporting the alignments with a single valid match

Let's map a couple of reads and get the output in SAM format

```
$ bowtie -a -v 2 --best --strata GENOMES/E_coli/bowtie_index/Ecoli_K12 -c  
GTTGGATCTGCCAAAAGAGCTGGCA,CAACATTATGAATGGCGAGGCAATCGCTGG,CCGAAAATGAAAGCCAGTA  
AAGAAG -S Ecoli_test.sam # output only the best matches for the "stratum" allowing for 2 mismatches
```

SAM format

Let's take a look at the resulting SAM file

```
$ less Ecoli_test.sam
```

- SAM stands for Sequence Alignment/Map format
- SAM stores data as human readable tab delimited file. Tab delimited meaning that [TAB] was used to separate columns. In Linux tab is coded as \t
- SAM stores read sequences, qualities, alignment coordinates, number of mismatches, number of times the read was mapped, was it mapped at all, etc.
- SAM file consists out of 2 sections: **Header section** that describes the reference sequences, source of data, method of alignments **Alignment section** describes each individual alignment
- BAM is a binary (machine code) version of sam. I would always recommend converting sam to bam and indexing resulting bam for speed and storage space considerations

SAM format

SAM header describes source of the data, reference sequence, method of the alignment etc. Each line starts with @ followed by 2 letter code, followed by tags and values

```
@HD VN:1.0 SO:unsorted
@SQ SN:NC_000913.3 LN:4641652
@PG ID:Bowtie VN:1.2 CL:"bowtie-align --wrapper basic-0 -a -v 2 --best --strata GENOMES/E_coli/bowtie_index/Ecoli_K12 -c GTTGGATCT
AAAGAGCTGGCA,CAACATTATGAATGGCGAGGCAATCGCTGG,CCGAAAATGAAAGCCAGTAAAGAAG -S Ecoli_test.sam"
```

- **@HD** header line **VN**:format version **SO**:sorting order
-
- **@SQ** reference info **SN**:reference sequence **LN**:sequence length **SP**:species
- **@RG** Read group **ID**:group id **CN**:sequencing centre **SM**:sample name
- **@PG** program **ID**:program name **VN**:program version **CL**:command

SAM format

Let's take a look at the body of a SAM containing alignments

```
0      0      NC_000913.3      4912      255      25M      *      0      0      GTTGGATCTGCCAAAAGAGCTGGCA      IIIIIIIIIIIIIIIIIIIIIII
A:i:0  MD:Z:25 NM:i:0
1      0      NC_000913.3      5408      255      30M      *      0      0      CAACATTATGAATGGCGAGGCAATCGCTGG      IIIIIIIIIIIIIIIIIIIIIII
A:i:0  MD:Z:30 NM:i:0
2      0      NC_000913.3      4639471 255      25M      *      0      0      CCGAAAATGAAAGCCAGTAAAGAAG      IIIIIIIIIIIIIIIIIIIIIII
A:i:0  MD:Z:25 NM:i:0
```

- SAM alignment format has 11 mandatory fields and any number of optional fields that may vary across different aligners
- This format was designed to contain all the information about each individual alignment and make it available for visualization

SAM format

Let's take a look at the body of a SAM containing alignments

Col	Field	Type	Description
1	QNAME	String	Read name (same as in FASTQ)
2	FLAG	Int	bitwise flag (encodes various metadata)
3	RNAME	String	Reference sequence name
4	POS	Int	1-based leftmost mapping position
5	MAPQ	Int	mapping quality
6	CIGAR	String	compact string that summarizes alignment
7	RNEXT	String	reference name for next mate (for PE reads)
8	PNEXT	Int	Position of the next mate (for PE reads)
9	TLEN	Int	observed template length (PE fragment length)
10	SEQ	String	read sequence
11	QUAL	String	read quality (Phred quality score)

SAM format

Bitwise flags

FLAG (in decimal)	Meaning
1	Read paired
2	Read mapped in proper pair
4	Read unmapped
8	Mate unmapped
16	Read reverse strand
32	Mate reverse strand
64	First in pair
128	Second in pair
256	Non primary alignment
512	Read fails vendor/platform quality check
1024	Read is PCR or optical duplicate
2048	Supplementary alignment

SAM format

Bitwise flags

- Bitwise flags are counter-intuitive and not easy to understand
- It was a way for the developers to include as much information into a single number as possible
- It is difficult to use flags correctly
- Individual flags are combined to form the value of the second column of a SAM file

For example, IF:

Read paired : 1

Read mapped in proper pair : 2

Mate reverse strand : 32

THEN the value of the second column will be **1 + 2 + 32 = 35** or **0X23** in hexadecimal, the software needs to decompose every value of the second column to select alignments with a certain attribute

SAM format

We can use ***samtools*** to take a look at bitwise flags

\$ samtools flags

Flags:

0x1	PAIRED	.. paired-end (or multiple-segment) sequencing technology
0x2	PROPER_PAIR	.. each segment properly aligned according to the aligner
0x4	UNMAP	.. segment unmapped
0x8	MUNMAP	.. next segment in the template unmapped
0x10	REVERSE	.. SEQ is reverse complemented
0x20	MREVERSE	.. SEQ of the next segment in the template is reversed
0x40	READ1	.. the first segment in the template
0x80	READ2	.. the last segment in the template
0x100	SECONDARY	.. secondary alignment
0x200	QCFAIL	.. not passing quality controls
0x400	DUP	.. PCR or optical duplicate
0x800	SUPPLEMENTARY	.. supplementary alignment

Explain SAM flags service: [Explain SAM Flags \(broadinstitute.github.io\)](https://broadinstitute.github.io/explain-sam-flags/)

SAM format

Manipulating SAM files with samtools

Let's check our theory that flag 35 means read paired ,properly paired, and mate in reverse

\$ `samtools flags 35`

- Samtools functions: viewing, filtering, sorting, indexing alignments, calling small nucleotide variants, data extraction, format conversion, duplicate read removal
- Samtools allows creation of Unix-style pipes, i.e. streaming the output of one command as input to another

Manual: <http://www.htslib.org/doc/samtools.html>

Let's create a more realistic SAM file to use for practice with **samtools**

Download a sample of M.tuberculosis sequencing file from GEO. Link to the file:

<https://trace.ncbi.nlm.nih.gov/Traces/sra/?run=SRR17227596>

SAM format

Manipulating SAM files with **samtools**

Download the sequencing reads from SRA archive

```
$ ~/programs/sratoolkit.3.0.1-ubuntu64/bin/prefetch -v -p SRR17227596
```

Check the contents of the newly creates directory, it should contain an .sra file

```
$ ls -lh SRR17227596
```

Dump the reads into fastq file

```
$ ~/programs/sratoolkit.3.0.1-ubuntu64/bin/fasterq-dump SRR17227596
```

Take a look at the file

```
$ head -n 20 SRR17227596.fastq
```

Print basic stats with **seqkit**

```
$ seqkit stats SRR17227596.fastq
```

SAM format

Manipulating SAM files with **samtools**

Subsample 200,000 random reads with seqtk

```
$ seqtk sample SRR17227596.fastq 200000 > Mtub_sub.fastq
```

Verify the results

```
$ seqkit stats Mtub_sub.fastq
```

Download M. tuberculosis H37Rv genome using **efetch**, the accession number is NC_000962.3

Name the resulting fasta file M_tuber_H37Rv.fasta

Move M. tuberculosis fasta file to **M_tuber/** directory in **GENOMES/**

Create a bowtie index for M. tuberculosis and place it in **bowtie_index/** folder inside in **M_tuber/** directory

Update README_GENOMES file accordingly

SAM format

Manipulating SAM files with **samtools**

Map the sub-sample to M. tuberculosis genome

```
$ bowtie --threads 2 -q --phred33-quals -n 1 -k 10 -l 28 --best  
GENOMES/M_tuber/bowtie_index/M_tuber_H37Rv Mtub_sub.fastq -S Mtub.sam # --threads 2 –  
use 2 processor cores in parallel; -k 10 – allow up to 10 alignments per reads; -q – reads file is fastq;  
--phred33-quals – Phred 33 quality scores; -n 1 – allow up to 1 mismatches in the seed; -l 28 –  
lengths of the seed is 28 nt at leftmost end of the read; --best – guarantees to find the matches
```

Let's take a look at the resulting SAM file

```
$ head Mtub.sam
```

```
$ wc -l Mtub.sam # What is the number of lines in the file, is it higher than the number of the  
reads?
```

Let's take a look a samtools manual and get familiar with the syntax and the options

<http://www.htslib.org/doc/samtools.html>

There is a lot we can do with samtools!

SAM format

Manipulating SAM files with **samtools**

A typical task in NGS analysis is to convert SAM files to BAM, sort, and index
BAM files are much smaller and faster to work with, while they contain the same data

```
$ samtools view -b Mtub.sam > Mtub.bam
```

```
$ head Mtub.bam # This produces gibberish, since bam files are not human readable
```

```
$ samtools sort Mtub.bam -o Mtub.sorted.bam # sort the alignments by chromosomal position
```

```
$ samtools index Mtub.sorted.bam # index the file for fast random access, lots of bioinformatics tools require bam files to be sorted and indexed
```

Compare the size of SAM and BAM files

```
$ ls -lh Mtub.sam | cut -d ' ' -f 5
```

```
$ ls -lh Mtub.bam | cut -d ' ' -f 5
```


SAM format

Manipulating SAM files with **samtools**

A typical task in NGS analysis is to convert SAM files to BAM, sort, and index
BAM files are much smaller and faster to work with, while they contain the same data

```
$ samtools view -b Mtub.sam > Mtub.bam
```

```
$ head Mtub.bam # This produces gibberish, since bam files are not human readable
```

```
$ samtools sort Mtub.bam -o Mtub.sorted.bam # sort the alignments by chromosomal position
```

```
$ samtools index Mtub.sorted.bam # index the file for fast random access, lots of bioinformatics tools require bam files to be sorted and indexed
```

Compare the size of SAM and BAM files

```
$ ls -lh Mtub.sam | cut -d ' ' -f 5
```

```
$ ls -lh Mtub.bam | cut -d ' ' -f 5
```

SAM format

Manipulating SAM files with **samtools**

Extract matches restricted to a genomic interval and write them into a new bam file

```
$ samtools view -b Mtub.sorted.bam "NC_000962.3:1000-10000" > region.bam
```

-b option ensures that extracted alignments are saved as bam, otherwise they are output as sam by default

Count all alignments (including unmapped) with **samtools view** and -c option

```
$ samtools view -c Mtub.sorted.bam
```

```
$ samtools view -c Mtub.sorted.bam "NC_000962.3:1000-10000"
```

-c can be combined with other options, like regions and filters

Filter alignments based in flags

Go to <http://broadinstitute.github.io/picard/explain-flags.html> and find a flag for unmapped reads (spoiler alert it is 4)

```
$ samtools view -b -F 4 Mtub.sorted.bam > Mtub.mapped.bam
```

-F 4 – print all alignments excluding unmapped (flag 4), output as bam (-b option)

```
$ samtools view -F 4 -c Mtub.sorted.bam
```

count all alignments (not reads!) for mapped reads

SAM format

Manipulating SAM files with **samtools**

Count all unmapped alignments (will be the same as unmapped reads)

```
$ samtools view -f 4 -c Mtub.sorted.bam
```

Count PCR/optical duplicates

```
$ samtools view -f 1024 -c Mtub.sorted.bam
```

Output SAM file statistics

```
$ samtools flagstat Mtub.sorted.bam
```

```
202575 + 0 in total (QC-passed reads + QC-failed reads)
0 + 0 secondary
0 + 0 supplementary
0 + 0 duplicates
164202 + 0 mapped (81.06% : N/A)
0 + 0 paired in sequencing
0 + 0 read1
0 + 0 read2
0 + 0 properly paired (N/A : N/A)
0 + 0 with itself and mate mapped
0 + 0 singletons (N/A : N/A)
0 + 0 with mate mapped to a different chr
0 + 0 with mate mapped to a different chr (mapQ>=5)
```

SAM format

Manipulating SAM files with **samtools**

Task: Use <https://broadinstitute.github.io/picard/explain-flags.html> site to figure out how to output forward and reverse alignments using **samtools** filtering based on flags

Hint: forward → check unmapped and reverse

Optional bowtie flags:

Take a look at the last 3 columns

```
$ samtools view Mtub.sorted.bam | cut -f 12-14 | head
```

```
XA:i:0 MD:Z:38G46 NM:i:1
XA:i:0 MD:Z:85 NM:i:0
XA:i:1 MD:Z:83G1 NM:i:1
XA:i:1 MD:Z:84A0 NM:i:1
XA:i:0 MD:Z:85 NM:i:0
XA:i:0 MD:Z:37G47 NM:i:1
```

SAM format

Manipulating SAM files with **samtools**

Bowtie optional fields

- **XA:i:<N>** - aligned read belongs to stratum N, the stratum is defined by the number of mismatches in the seed regions
- **MD:Z:<S>** - <S> is a string representation of mismatches in the reference, for example MD:Z:38G46 means that 38 leftmost bases match, then base 39 is mismatch with G in the reference, and 46 next bases match
- **NM:i:<INT>** - Contains edit distance to reference stored as integer value <INT>. Edit distance is measure of similarity between sequences, it tells as how many edits we need to make to

Optional field specifications: <https://samtools.github.io/hts-specs/SAMtags.pdf>

SAM format

Manipulating SAM files with **samtools**

We can filter the alignments based on the optional fields

Example, count the number of alignments with 0 mismatches

```
$ samtools view Mtub.sorted.bam | grep 'XA:i:0' | wc -l
```

How to get detailed help for samtools

```
$ samtools view -?
```

```
$ samtools sort -?
```

Using multiple processor cores with -@ option

```
$ time samtools sort Mtub.sam -o Mtub.sorted.bam # time the sorting time with time
```

```
$ time samtools -@ 2 sort Mtub.sam -o Mtub.sorted.bam # -@ 2 tells to use 2 processor cores
```

Mapping paired-end reads with Bowtie

Download practice paired-end sample

To practice, we will download one sequencing samples from the following paper

[Front Microbiol.](#) 2019; 10: 309.

PMCID: PMC6399466

Published online 2019 Feb 26. doi: [10.3389/fmicb.2019.00309](https://doi.org/10.3389/fmicb.2019.00309)

PMID: [30863380](https://pubmed.ncbi.nlm.nih.gov/30863380/)

Whole Genome Sequencing of *Mycobacterium tuberculosis* Clinical Isolates From India Reveals Genetic Heterogeneity and Region-Specific Variations That Might Affect Drug Susceptibility

[Jayshree Advani](#),^{1,2,3,†} [Renu Verma](#),^{1,†} [Oishi Chatterjee](#),^{1,2,4} [Praveen Kumar Pachouri](#),⁵
[Prashant Upadhyay](#),⁵ [Rajesh Singh](#),⁵ [Jitendra Yadav](#),⁵ [Farah Naaz](#),⁵ [Raju Ravikumar](#),⁶
[Shashidhar Buggi](#),^{7,8,‡} [Mrutyunjay Suar](#),⁹ [Umesh D. Gupta](#),⁵ [Akhilesh Pandey](#),^{1,3,10,11,12,13}
[Devendra S. Chauhan](#),⁵ [Srikanth Prasad Tripathy](#),^{5,*‡} [Harsha Gowda](#),^{1,3,*} and
[T. S. Keshava Prasad](#)^{1,2,*}

Link to the paper: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6399466/>

Mapping paired-end reads with Bowtie

Download practice paired-end sample

- Scan the methods section of the paper to understand what was done
- Locate data availability section with BioProject accession number
- Use the accession number to search NCBI SRA database
- Locate and download one of the paired-end sequencing samples we will practice with. You can use **Entrez utilities** or **wget**

Hints:

How to search for SRA accession number with BioProject accession

```
$ esearch -db sra -query <PRJNAxxxxxx> | efetch -format runinfo # this will print RunInfo file
```

How to fetch SRA file

```
$ ~/programs/sratoolkit.3.0.1-ubuntu64/bin/prefetch -v -p <SRRxxxxxxx>
```


Mapping paired-end reads with Bowtie

Download practice paired-end sample

Hints:

We can also download SRA files with **wget**, URLs can be found in RunInfo file

`$ esearch -db sra -query PRJNA379070 | efetch -format runinfo | cut -d ',' -f 1,10` # This command will grab the SRA accession numbers and their URLs from RunInfo file

We can use SRA file URL to download a file directly with **wget**.

`$ wget https://sra-downloadb.be-md.ncbi.nlm.nih.gov/sos5/sra-pub-zq-14/SRR005/341/SRR5341272.sralite.1` # there is a possibility that SRA will block your IP if you are downloading too many files

Resulting file can be dumped to FASTQ using **fasterq-dump** from **sra-tools**

`$ ~/programs/sratoolkit.3.0.1-ubuntu64/bin/fasterq-dump --split-3 SRR5341272.sralite.1` # --split-3 – option will split paired-end SRA into 2 properly formatted fastq files from Read1 and Read2

Now, use these commands to retrieve one of the sequencing samples from Advani et al, 2019, rename fastq files to **india.R1.fastq** and **india.R2.fastq**

Mapping paired-end reads with Bowtie

Let's view the basic stats for these fastq files and map the reads

```
$ seqkit -j 2 stats india.R* -a # -j 2 will use 2 processor cores, -a will print additional columns in summary stats
```

To reduce mapping types, let's subsample 10,000 random reads

```
$ seqtk sample -s100 india.R1.fastq 10000 > sub1.fq
```

```
$ seqtk sample -s100 india.R2.fastq 10000 > sub2.fq # when sub-sampling paired-end reads we need to set the same seed for random number algorithm for both commands
```

Map paired-end reads to M.tuberculosis genome

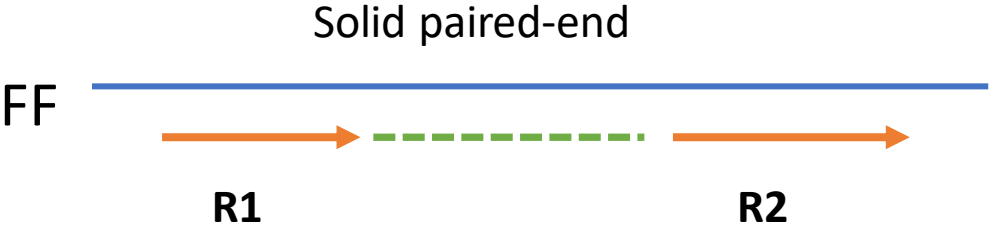
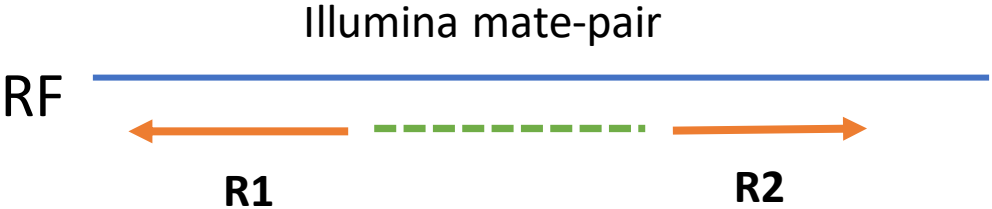
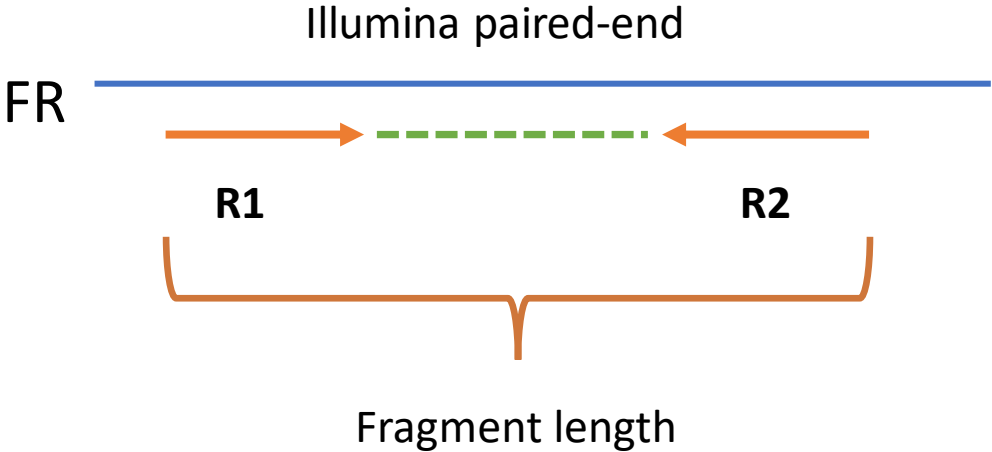
```
$ bowtie --threads 2 --fr GENOMES/M_tuber/bowtie_index/M_tuber_H37Rv -1 sub1.fq -2 sub2.fq -S india.sam # --fr — is a paired-end specific option that tells the aligners that mate 1 is in forward direction and mate 2 is mapped in reverse. Note, the low number of mapped reads, bowtie is not a suitable aligner in this case!
```

What happens if we specify the **strandness** of the library as --rf?

```
bowtie --threads 2 --rf GENOMES/M_tuber/bowtie_index/M_tuber_H37Rv -1 sub1.fq -2 sub2.fq test.aln # Almost no reads are mapped, specifying strandness is important
```

Mapping paired-end reads with Bowtie

Typed of sequencing library strandness



Mapping paired-end reads with Bowtie

Let's take a look at the sam file format after paired-end alignments

View basic stats with samtools

```
$ samtools flagstat india.sam
```

Let's take a look at some lines of the SAM file

```
$ samtools view -h india.sam | head # -h – option will skip the header lines; most of the lines in this case are not aligned hence fields storing coordinates are 0
```

Let's retain mapped reads only and take a look at paired-end specific columns 7, 8, 9

```
$ samtools view -F 4 india.sam | head # take a look at mapped reads
```

```
$ samtools view -F 4 india.sam | head | cut -f 7-9 # select columns specific to paired-end reads
```

Column	SAM Description	Meaning
7	RNEXT	Reference name for the second mate alignment
8	PNEXT	Coordinate for the start for second mate alignment
9	TLEN	Fragment length

Mapping with Bowtie2

Install bowtie2, take a look at bowtie2 documentation

Bowtie2 paper: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3322381/>

[Nat Methods](#). Author manuscript; available in PMC 2013 Apr 1.

PMCID: PMC3322381

Published in final edited form as:

NIHMSID: NIHMS366740

[Nat Methods](#). 2012 Mar 4; 9(4): 357–359.

PMID: [22388286](#)

Published online 2012 Mar 4. doi: [10.1038/nmeth.1923](https://doi.org/10.1038/nmeth.1923)

Fast gapped-read alignment with Bowtie 2

[Ben Langmead](#)^{1,2} and [Steven L Salzberg](#)^{1,2,3}

Scan through bowtie2 alignments and try to understand software capabilities and options:

<https://bowtie-bio.sourceforge.net/bowtie2/manual.shtml>

Mapping with Bowtie2

Key properties of bowtie2

1. Bowtie2 allows gaps
2. Bowtie2 is faster and more sensitive than bowtie for reads < 50 bp
3. Supports both end-to-end and local alignments, bowtie only had end-to-end alignments
4. Can map long reads, over 1000 bp
5. Bowtie2 does have “strata”, the alignments have continuous mapping quality scores. Bowtie had only 2 values: 0 and 255 as mapping quality scores
6. In case of paired-end alignments, bowtie2 tries to find valid matches for individual pairs

Mapping with Bowtie2

Difference between **end-to-end** and **local** alignments

- End-to-end alignment searches for matches that include all characters of a sequence.
- By default, bowtie2 preforms end-to-end alignment

End-to-end alignment example:

Read: GACTGGGCGATCTCGACTTCG

Reference: GACTGCGATCTCGACATCG

Alignment:

Read: GACTGGGCGATCTCGACTTCG

||||| ||||| |||

Reference: GACTG--CGATCTCGACATCG

Mapping with Bowtie2

Difference between **end-to-end** and **local** alignments

In “**local**” alignment, some characters at the end of the read may not participate, bases at the start or end of a read that are omitted are called “**soft-clipped**”

Local alignments are invoked with **--local** option

Example of a local alignment

Read: ACGGTTGCGTTAATCCGCCACG

Reference: TAACTTGCGTTAAATCCGCCTGG

Alignment:

Read: ACGGTTGCGTTAA-TCCGCCACG

 ||||| |||||

Reference: TAACTTGCGTTAAATCCGCCTGG

Mapping with Bowtie2

Bowtie2 alignment scoring

- Alignment score measures the similarity between aligned sequences
- Higher score means higher similarity
- In end-to-end mode the scores are calculated by subtracting penalties for gaps and mismatches, and in local alignments by adding “bonuses” for matches
- End-to-end scoring scheme: mismatch at high quality base is **-6**; gap open **-5**, gap extend **-3** and second gap extend **-3**. The best possible match quality is 0.
- In local alignments we have the same mismatch and gap penalties as in end-to-end mode, however, every match receives **+2** bonus. The score a perfectly matching reads will **2 * read_length**
- The scoring schemes can be changes with corresponding options, however default schemes work the best in most cases

Mapping with Bowtie2

Bowtie2 mapping quality

- Higher mapping quality corresponds to higher uniqueness
- A read can have equally good matches to multiple regions, and we have no basis to prefer one match to another
- A preferred alignment can be selected based on mapping quality which is a measure of probability that a read may have a different point of origin than the one determined by a given match
- Mapping quality is a non-negative integer $Q = -10 \log p$, where p is an estimate of probability that the read can have a true origin in a different location
- Mapping quality depends on the difference between the best and a second-best alignment, the bigger is this difference, the higher is the mapping quality and a “uniqueness” of this match
- For example, a quality of 10 or less means that there is 1 in 10 probability that the read originated some other locus than a given location of a match. Mapping qualities are recorded in MAPQ column of a SAM file (column 5).

Mapping with Bowtie2

Assumptions in paired-end mapping

- In case of paired-end libraries we sequence a DNA fragment from both ends resulting in mate 1 and mate 2 reads.
- There are certain expectations as to how both mate reads are supposed to match the reference based on the library preparation protocol.
- For example, in first-stranded (FR) libraries mate 1 originates from the forward strand of the reference and mate 2, from the reverse.
- In most of library construction protocols, we expect 200 – 500 interval between mates
- **Concordant** alignments match these expectations, while **discordant** do not
- An example of discordant pair would be mate 1 mapping 2000 bp away from mate 2

Mapping with Bowtie2

Practicing **bowtie2** alignments

Start with creating a bowtie2 index using M. tuberculosis genome

Create bowtie2 index and place it in **/GENOMES/M_tuber/bowtie2** directory

```
$ bowtie2-build ../M_tuber_H37Rv.fasta M_tuber_H37Rv
```

Bowtie2 syntax

```
bowtie2 [options] -x <bt2-idx> {-1 <m1> -2 <m2> | -U <r>} -S <sam>
```

Run a single-end alignment with bowtie2

```
$ bowtie2 -p 2 --sensitive -x GENOMES/M_tuber/bowtie2_index/M_tuber_H37Rv -U sub1.fq -S  
test_SE.sam # Take a look at the summary of alignments printed to screen
```

Now, let's try to map paired-end reads

```
$ bowtie2 -p 2 --fr --sensitive -x GENOMES/M_tuber/bowtie2_index/M_tuber_H37Rv -1 sub1.fq -2  
sub2.fq -S test_PE.sam 2> align_stats.txt # We can re-direct alignment statistics to file from StdErr
```

Mapping paired-end reads with Bowtie2

Flagstat output for paired alignments

Let's take a look at the output of **samtools flagstat**

`$ samtools flagstat test_PE.sam`

```
20000 + 0 in total (QC-passed reads + QC-failed reads)
0 + 0 secondary
0 + 0 supplementary
0 + 0 duplicates
19528 + 0 mapped (97.64% : N/A)
20000 + 0 paired in sequencing
10000 + 0 read1
10000 + 0 read2
18132 + 0 properly paired (90.66% : N/A)
19312 + 0 with itself and mate mapped
216 + 0 singletons (1.08% : N/A)
0 + 0 with mate mapped to a different chr
0 + 0 with mate mapped to a different chr (mapQ>=5)
```

samtools flagstat manual: <http://www.htslib.org/doc/samtools-flagstat.html>

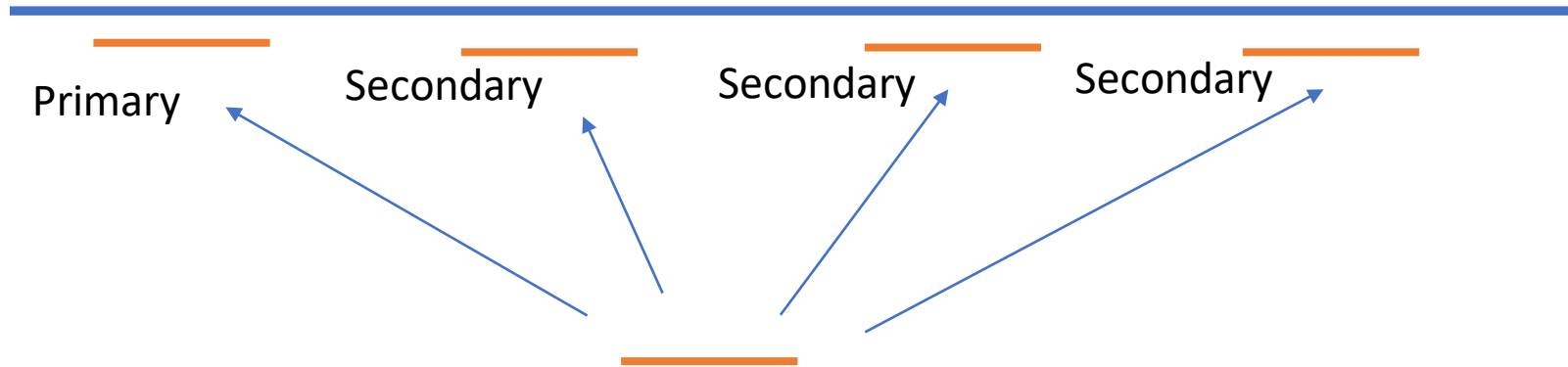
Explain SAM flags: <https://broadinstitute.github.io/picard/explain-flags.html>

Mapping paired-end reads with Bowtie2

Flagstat output for paired alignments

Some cryptic terms in **flagstat** output:

Secondary alignments – refers to multimapping reads, one of the mutimappers is considered “primary”, and the rest are secondary alignments

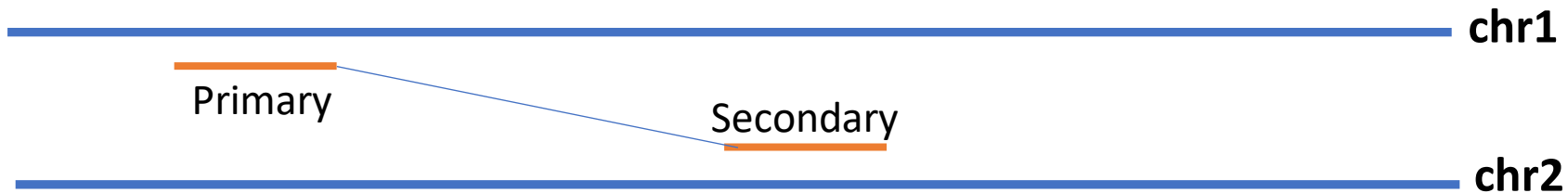


Mapping paired-end reads with Bowtie2

Flagstat output for paired alignments

Some cryptic terms in **flagstat** output:


Supplementary alignments – are related to non-linear/fusion/chimeric alignments, where the sequence is broken into segments to align. One of the segment will be considered “primary” and another “supplementary”. This choice can be arbitrary



Mapping with BWA

Install BWA (Burrows-Wheelers Aligner) and take a look at the manual

Fast and accurate short read alignment with Burrows–Wheeler transform

Heng Li, Richard Durbin  [Author Notes](#)

Bioinformatics, Volume 25, Issue 14, July 2009, Pages 1754–1760,

<https://doi.org/10.1093/bioinformatics/btp324>

Published: 18 May 2009 **Article history** ▼

 PDF  Split View  Cite  Permissions  Share ▼

Original paper - <https://academic.oup.com/bioinformatics/article/25/14/1754/225615>

BWA mem paper - <https://arxiv.org/pdf/1303.3997v2.pdf>

Manual: <https://bio-bwa.sourceforge.net/bwa.shtml>

Mapping with BWA

Different algorithms within BWA

BWA employs 3 different algorithms: BWA-backtrack, BWA-MEM, and BWA-SW

The syntax of alignments for all 3 algorithms

```
bwa aln ref.fa short_read.fq > aln_sa.sai # BWA-backtrack, best for short reads below 100 bp
```

```
bwa mem ref.fa reads.fq > aln-se.sam # BWA-MEM single-end
```

```
bwa mem ref.fa read1.fq read2.fq > aln-pe.sam # BWA-MEM paired-end
```

```
bwa samse ref.fa aln_sa.sai short_read.fq > aln-se.sam # converts sai file to sam for single-end reads
```

Mapping with BWA

Different algorithms within BWA

```
bwa sampe ref.fa aln_sa1.sai aln_sa2.sai read1.fq read2.fq > aln-pe.sam #
```

converts sai file to sam for paired-end reads

```
bwa bwasw ref.fa long_read.fq > aln.sam # BWA SW algorithm for reads longer than 100
```

bp, now superseded by **BWA MEM**

Mapping with BWA

Different algorithms within BWA

- BWA-backtrack invoked with **bwa aln** command is recommended for short reads below 100 bp
- BWA SW (**bwa samse** and **bwa sampe**) commands are now deprecated, the author of BWA recommends using BWA-MEM instead
- BWA MEM is current version of the algorithm that should be used for the reads longer than 100 bp
 - ✓ Backtrack algorithm searches for exact end-to-end matches. If an exact match is not found, the program tries all single edits, all possible 2-character edits, etc.
 - ✓ This is problematic since the time of execution grows exponentially with increasing number of errors and, I expect, increasing read length. To avoid this the program is instructed to stop and throw out a read once the divergence between the query and reference increases over a certain threshold set in -**maxDiff** option
 - ✓ BWA-MEM detects long exact matches between the read and the reference, and chains them into local or global alignments, based on what is more appropriate in that specific case

Mapping with BWA

Practice mapping with BWA

First, create an index of M. tuberculosis genome in **GENOMES/M_tuber/bwa_index**

```
$ bwa index ../M_tuber_H37Rv.fasta
```

Let's map the reads with BWA-backtrack algorithm

```
$ bwa aln GENOMES/M_tuber/bwa_index/M_tuber_H37Rv.fasta sub1.fq > sub1_sa.sai
```

Now we need to convert an intermediate **.sai** file to **.sam**

```
$ bwa samse GENOMES/M_tuber/bwa_index/M_tuber_H37Rv.fasta sub1_sa.sai sub1.fq > sub1.sam
```

```
$ samtools flagstat sub1.sam # take a look at mapping stats
```

Let's map paired-end reads with **bwa aln**

```
$ bwa aln GENOMES/M_tuber/bwa_index/M_tuber_H37Rv.fasta sub1.fq > sub1_sa.sai
```

```
$ bwa aln GENOMES/M_tuber/bwa_index/M_tuber_H37Rv.fasta sub2.fq > sub2_sa.sai
```

```
$ bwa sampe GENOMES/M_tuber/bwa_index/M_tuber_H37Rv.fasta sub1_sa.sai sub2_sa.sai sub1.fq  
sub2.fq > sub_pe.sam
```

```
$ samtools flagstat sub_pe.sam
```

Mapping with BWA

Practice mapping with BWA

Map single end reads with bwa mem

```
$ bwa mem GENOMES/M_tuber/bwa_index/M_tuber_H37Rv.fasta sub1.fq > mem_se.sam
```

```
$ samtools flagstat mem_se.sam
```

Map paired end reads with bwa mem

```
$ bwa mem GENOMES/M_tuber/bwa_index/M_tuber_H37Rv.fasta sub1.fq sub2.fq > mem_pe.sam
```

```
$ samtools flagstat mem_pe.sam
```

Take a look at the alignments

```
$ samtools view -h mem_pe.sam
```

```
$ samtools view mem_pe.sam | head | cut -f 5 # BWA produces proper MAPQ scores (mapping quality)  
recorded in column 5 of the SAM file
```

Visualizing alignments

Let's create a larger subsample from **india.Rx.fastq** files, this time will sample 200,000 files

```
$ seqtk sample -s100 india.R1.fastq 200000 > sub1.fq
```

```
$ seqtk sample -s100 india.R2.fastq 100000 > sub2.fq
```

Verify that the number of sequences is the same

```
$ grep '@SRR' sub1.fq | wc -l
```

```
$ grep '@SRR' sub1.fq | wc -l
```

Map the reads to M. tuberculosis genome with BWA

```
$ bwa mem GENOMES/M_tuber/bwa_index/M_tuber_H37Rv.fasta sub1.fq sub2.fq > mem_pe.sam
```

Convert resulting file to bam, sort and index

```
$ samtools view -b mem_pe.sam > mem_pe.bam
```

```
$ samtools sort mem_pe.bam -o mem_pe.sorted.bam
```

```
$ samtools index mem_pe.sorted.bam
```

Visualizing alignments

View the alignments with **samtools tview**

Manual: <http://www.htslib.org/doc/samtools-tview.html>

samtools tview is a fast text-based interactive browser, that allows to view the alignments in Linux terminal

Command syntax

```
samtools tview -p <chr:start-end> -d display -s STR <sorted.bam>  
<reference.fasta>
```

We need to know the name of the chromosomes (or a single chromosome in our case), to navigate across the genome

```
$ head GENOMES/M_tuber/M_tuber_H37Rv.fasta
```

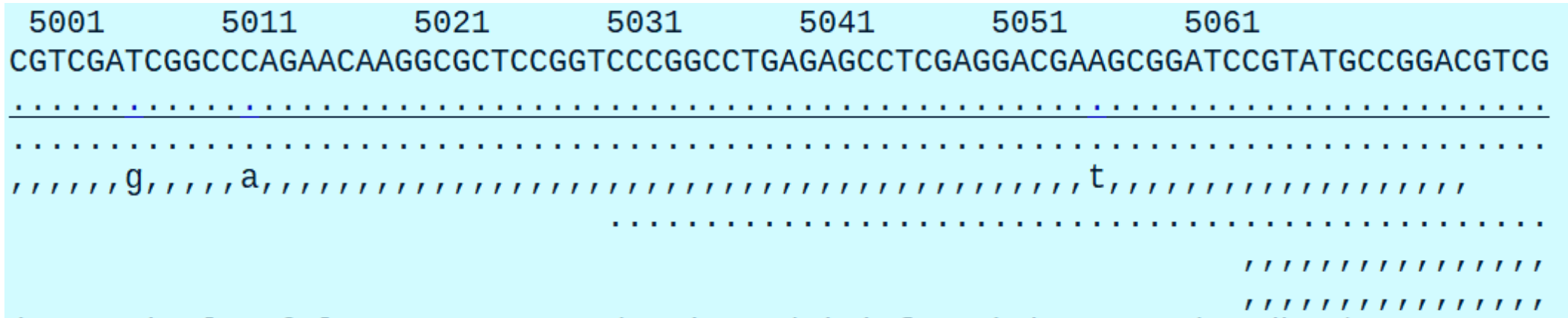
```
>NC_000962.3 Mycobacterium tuberculosis H37Rv, complete genome  
TTGACCGATGACCCCGGTTCAGGCTTCACCACAGTGTGGAACGCGGTCGTCTCCGAACTTAAC  
CTAAGGTTGACGACGGACCCAGCAGTGATGCTAATCTCAGCGCTCCGCTGACCCCTCAGCAAA  
GCTCAATCTCGTCCAGCCATTGACCATCGTCGAGGGGTTTGCTCTGTTATCCGTGCCGAGCAG  
CAAAACGAAATCGAGCGCCATCTGCGGGCCCCGATTACCGACGCTCTCAGCCGCCGACTCGGA
```

Visualizing alignments

View the alignments with **samtools tview**

Let's view the alignments 5000 bp

```
$ samtools tview -p NC_000962.3:5000 mem_pe.sorted.bam  
GENOMES/M_tuber/M_tuber_H37Rv.fasta
```



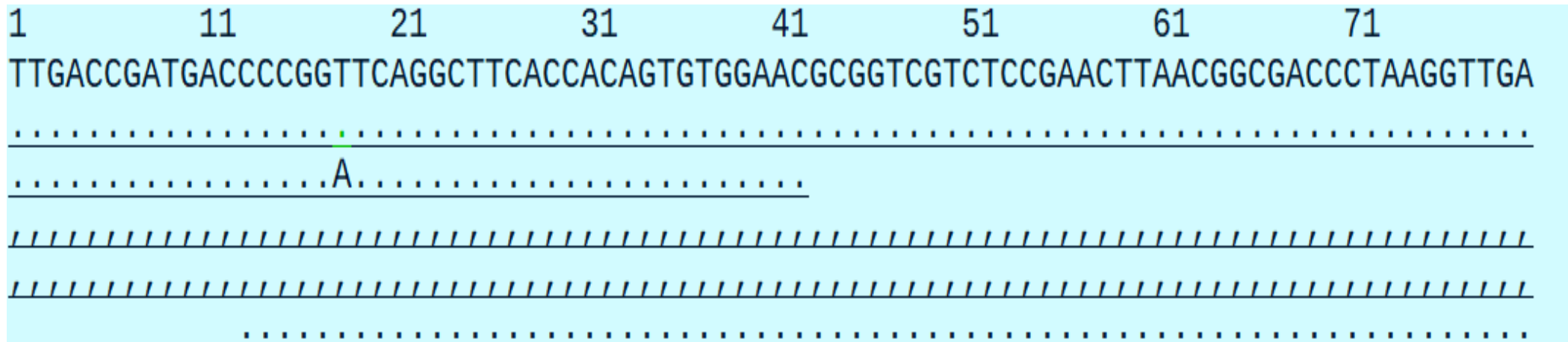
- Top line is a reference sequence
- Below the reference sequence we see a consensus sequence in dot notation. In dot notation, '.' are characters matching the reference, ',' matching the reference in reverse, only mismatches are shown as characters

Visualizing alignments

View the alignments with **samtools tview**

Let's view the alignments 5000 bp

```
$ samtools tview -p NC_000962.3:5000 mem_pe.sorted.bam  
GENOMES/M_tuber/M_tuber_H37Rv.fasta
```



- Here we observe a T/A mismatch highlighted in green
- Sequence of dots under the reference is a consensus sequence – theoretical representation of sequence of where each character is the most frequent character observed in aligned reads

Visualizing alignments

View the alignments with **Integrative Genomic Viewer (IGV)**

Home page: <https://software.broadinstitute.org/software/igv/>

User guide: <https://software.broadinstitute.org/software/igv/UserGuide>

- IGV is a powerful genomic viewer designed in Broad Institute
- IGV has a graphical user interface
- It can show alignments, coverage data, genomic intervals, genomic variants and other forms of genomic data
- In addition, IGV's functionality permits data transformation and export

Visualizing alignments

View the alignments with **Integrative Genomic Viewer (IGV)**


Let's obtain gene annotation for M. tuberculosis H37Rv with the accession number NC_000962.3

Genome

Genome ▾

NC_000962.3 |

Create alert Limits Advanced



In June 2023, Genome record pages will be redirected to the new [Datasets Taxonomy page](#). [Learn more](#)

Mycobacterium tuberculosis

Reference genome: **Mycobacterium tuberculosis H37Rv**

Download sequences in FASTA format for **genome**, **protein**

Download genome annotation in **GFF**, **GenBank** or **tabular** format

BLAST against Mycobacterium tuberculosis **genome**, **protein**

All 7023 genomes for species:

We need GFF file containing annotation data for NC_000962.3 genome

Visualizing alignments

View the alignments with **Integrative Genomic Viewer (IGV)**

Create **annotation/** directory in **GENOMES/M_tuber** folder

```
$ mkdir -p GENOMES/M_tuber/annotation
```

Download GFF file using **wget**

```
$ wget
```

```
https://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/195/955/GCF\_000195955.2\_ASM19595v2/GCF\_000195955.2\_ASM19595v2\_genomic.gff.gz
```

Unzip GFF file and move it to **GENOMES/M_tuber/annotation**

```
$ gunzip GCF_000195955.2_ASM19595v2_genomic.gff.gz && mv  
GCF_000195955.2_ASM19595v2_genomic.gff GENOMES/M_tuber/annotation/
```

Visualizing alignments

Create reference fasta index

Reference index is created using **samtools faidx** command

Read the manual: <http://www.htslib.org/doc/samtools-faidx.html>

Command syntax

```
samtools faidx <ref.fasta>
```

Create index

```
$ cd GENOMES/M_tuber
```

```
$ samtools faidx M_tuber_H37Rv.fasta
```

```
$ ls -lh
```

We will see a newly created ***M_tuber_H37Rv.fasta.fai***

```
$ cat M_tuber_H37Rv.fasta.fai # the output is a 5 column text file, like below
```

```
NC_000962.3 4411532 63 70 71
```

Visualizing alignments

Create reference fasta index

Format of the *.fai* file

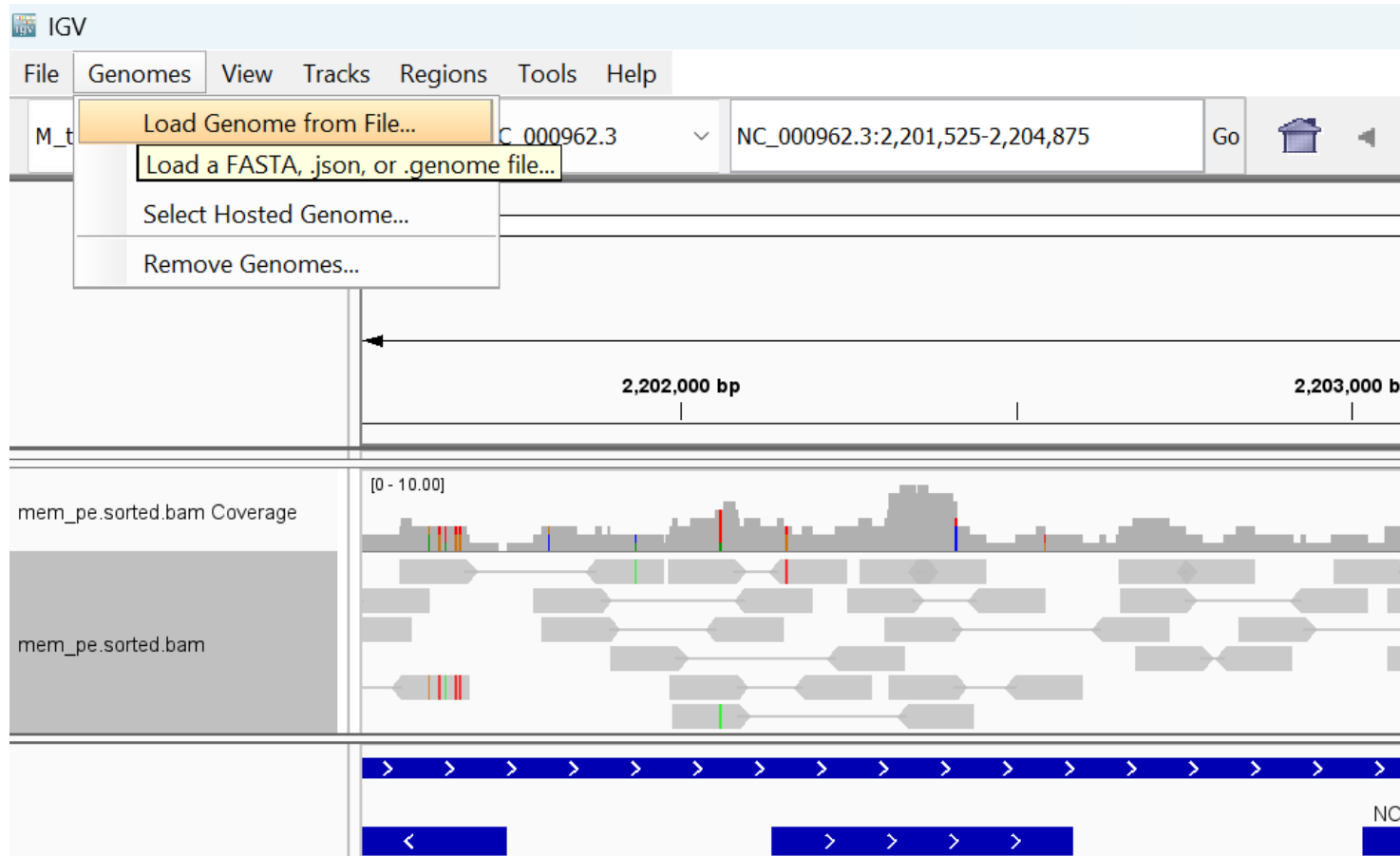
- Column 1: The contig name. In your FASTA file, this is preceded by '>'
- Column 2: The number of bases in the contig
- Column 3: The byte index of the file where the contig sequence begins. (Notice how it constantly increases by roughly the amount in column 2?)
- Column 4: **bases** per line in the FASTA file
- Column 5: **bytes** per line in the FASTA file

We can also use **samtools faidx** to extract subsequences from a fasta file:

```
$ samtools faidx M_tuber_H37Rv.fasta NC_000962.3:2000-3000 # this will extract a sequence  
between 2000 and 3000 bp
```

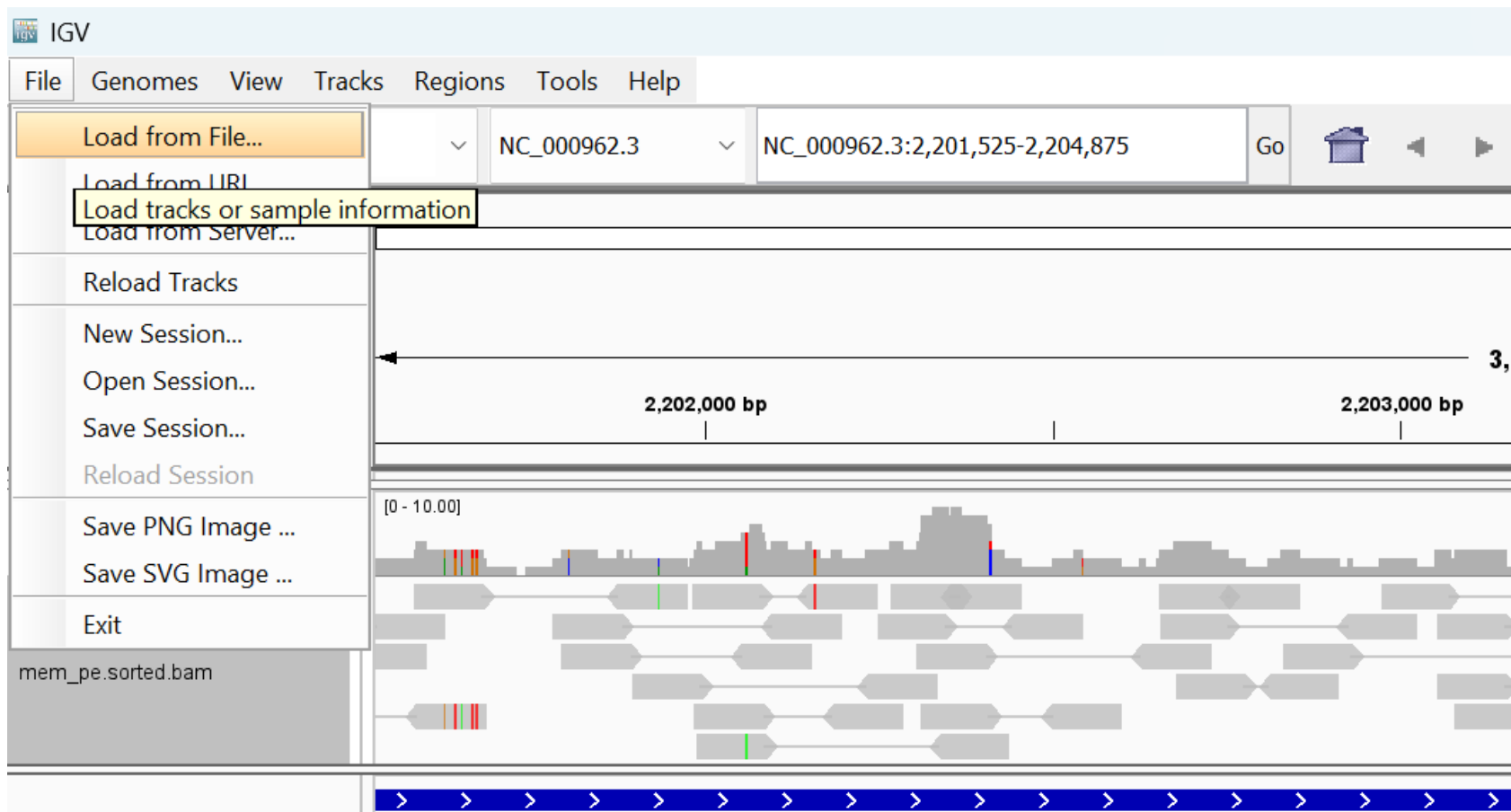
Visualizing alignments

Launch IGV and load the genome reference from file



Visualizing alignments

Load GFF and bam file using “**Load from File**” tab, bam file needs to be sorted and indexed



Visualizing alignments

Now we can expand the tracks for better view, play around with IGV to explore it's interface and capabilities

