HISAT2

graph-based alignment of next generation sequencing reads to a population of genomes

Search

- Main
- About
- Manual
- HISAT-3N
- Download
- HowTo
- Links

HISAT2 is a fast and sensitive alignment program for mapping next-generation sequencing reads (both DNA and RNA) to a population of human genomes as well as to a single reference genome. Based on an extension of BWT for graphs (Sirén et al. 2014), we designed and implemented a graph FM index (GFM), an original approach and its first implementation. In addition to using one global GFM index that represents a population of human genomes, **HISAT2** uses a large set of small GFM indexes that collectively cover the whole genome. These small indexes (called local indexes), combined with several alignment strategies, enable rapid and accurate alignment of sequencing reads. This new indexing scheme is called a Hierarchical Graph FM index (HGFM).

The HISAT-3N paper published at *Genome Research*. 7/1/2021

HISAT-3N beta release 12/14/2020

HISAT-3N is a software system for analyzing nucleotide conversion sequencing reads. See the HISAT-3N for more details.

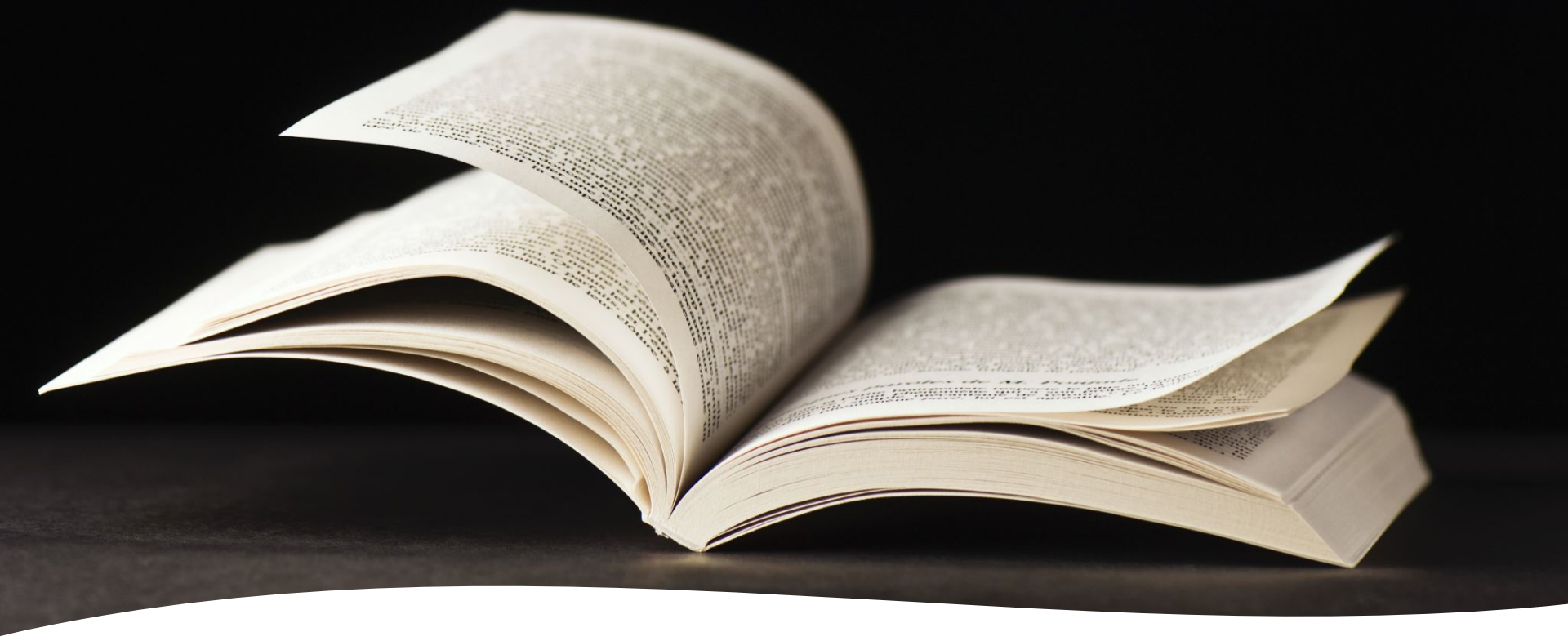
Mapping with HISAT2

Outline

- **Class Activity #1, ~5 minutes**
- Lecture for ~25 minutes
- Class Activity #2, ~15 minutes

ALIGNER	Designed for (Type of Data)	Programming Language	Algorithm	Mode	Clip
BOWTIE2	DNA, RNA	C++	BWT	Local	Soft
HISAT2	DNA, RNA	Python	BWT	Local	Soft
STAR	RNA	C++	BWT	Local	Soft
TOPHAT2	RNA	C++	BWT	Global	Soft

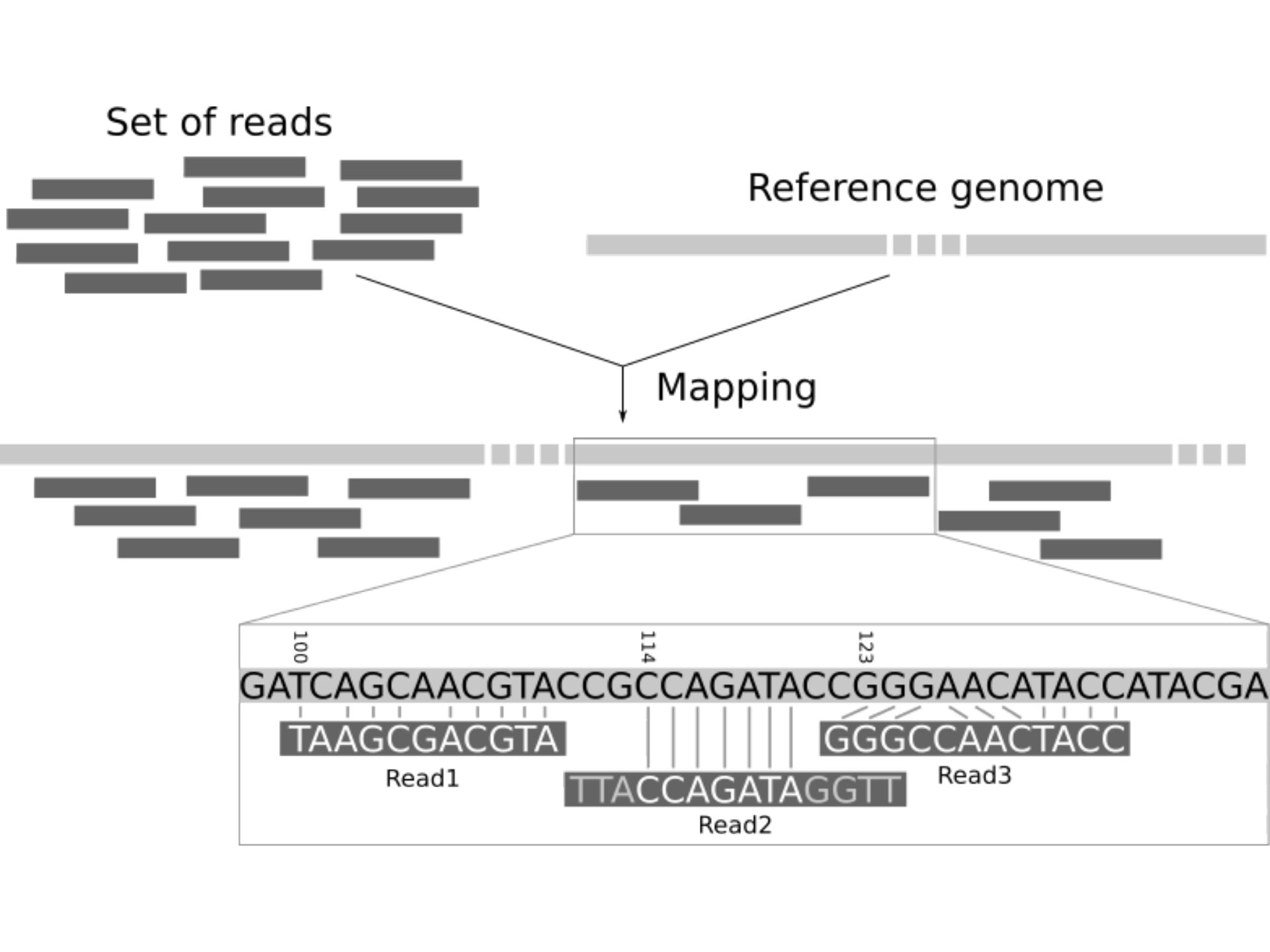
BWT = Burrows-Wheeler transform

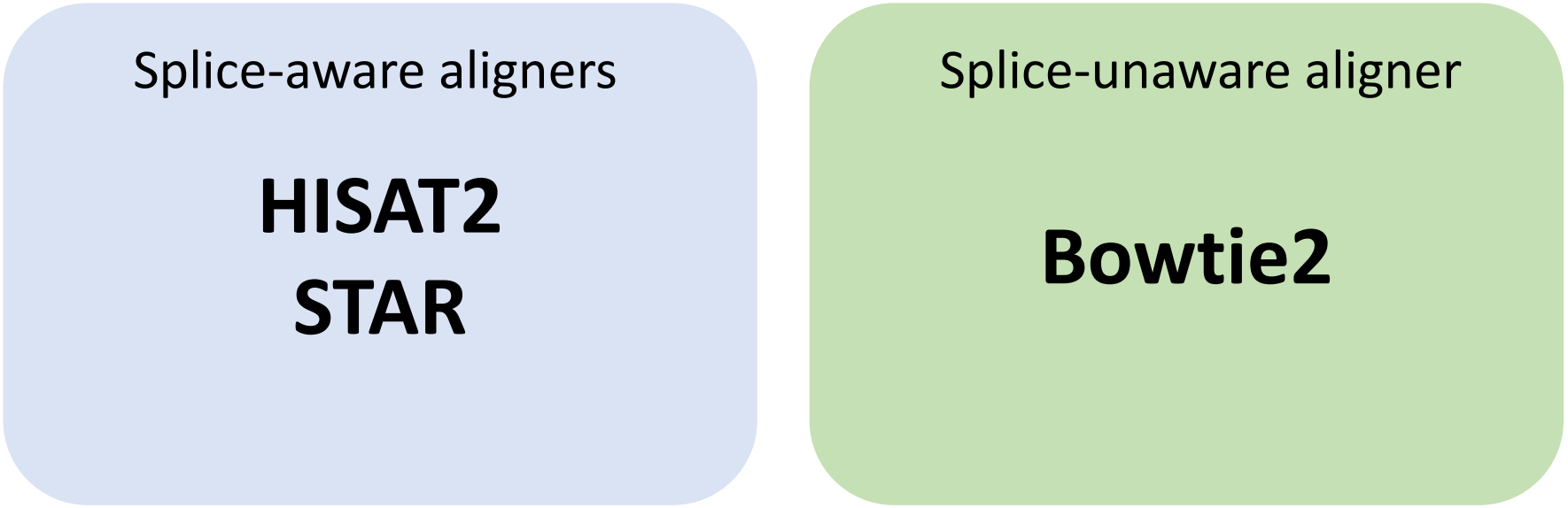


Indexing benefits

Indexing a genome can be explained **similar to indexing a book**.

If you want to know on which page a chapter begins, it is much more efficient to look up the page number in a pre-built index (table of contents) than going through every page of the book until you found the chapter you are looking for.


Set of reads
Reference genome
Mapping
100
114
123
GATCAGCAACGTACCGCCAGATACCGGGGAACATACCATACGA
TAAGCGACGTA
Read1
TTACCAGATAGGTT
Read2
GGGCCAACTACC
Read3



RNA-Seq

ChIP-Seq

What does it mean to be splice-aware?

- The major problem is that introns not only vary in length but that they can also be very long.

- A splice-unaware aligner will have to introduce a **long gap** in the mapping of a read to span an intron.

- Would know not to try to align reads to introns, and identifies exons and tries to align to those instead, ignoring introns altogether

Two groups of splice-aware aligners

- **Group 1:** Splice-aware aligners that use the genome sequence and known gene annotations to calculate gene or transcript abundance – these ***can not*** be used to identify new splice junctions

- **Group 2:** *de novo* splice-aware aligners which can align RNA-Seq reads to a reference genomic sequence without prior information on gene annotations
  - STAR (Spliced Transcripts Alignment to a Reference)

HISAT2

- Stands for **h**ierarchical **i**ndexing for **s**pliced **a**lignment of **t**ranscripts 2
- HISAT2 is an aligner that is used for mapping next-generation sequencing reads
  - Used for whole genome, whole-exome, and transcriptome datasets

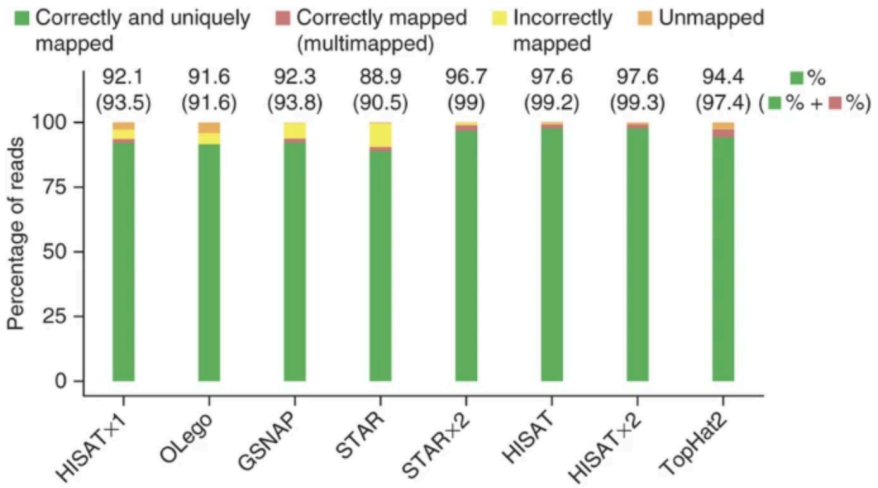
HISAT2 is fast

- Is the fastest spliced mapper currently available
- This is possible due to its novel indexing strategy
- Uses BWT algorithm to compress genomes therefore, requiring very little memory to store.
- But additionally, it builds these genomes using FM-indexing.
  - Using **both** makes it possible to search through genomes rapidly

HISAT2 has a small memory footprint

- The STAR program runs faster than TopHat2 but both have a memory requirement of ~28GB
- The memory requirement for HISAT2 is ~5GB
  - This makes it possible to do alignments on your laptop

Figure 3: Alignment accuracy of spliced alignment software for 20 million simulated 100-bp reads.



The dataset

nature COMMUNICATIONS

ARTICLE

https://doi.org/10.1038/s41467-021-26159-1 OPEN

Tcf1 and Lef1 provide constant supervision to mature CD8$^+$ T cell identity and function by organizing genomic architecture

Qiang Shan[1,5], Xiang Li[2,5], Xia Chen[3], Zhouhao Zeng[2], Shaoqi Zhu[2], Kexin Gai[1], Weiqun Peng[2✉] & Hai-Hui Xue[1,4✉]

T cell identity is established during thymic development, but how it is maintained in the periphery remains unknown. Here we show that ablating Tcf1 and Lef1 transcription factors in mature CD8$^+$ T cells aberrantly induces genes from non-T cell lineages. Using high-throughput chromosome-conformation-capture sequencing, we demonstrate that Tcf1/Lef1 are important for maintaining three-dimensional genome organization at multiple scales in CD8$^+$ T cells. Comprehensive network analyses coupled with genome-wide profiling of chromatin accessibility and Tcf1 occupancy show the direct impact of Tcf1/Lef1 on the T cell genome is to promote formation of extensively interconnected hubs through enforcing chromatin interaction and accessibility. The integrative mechanisms utilized by Tcf1/Lef1 underlie activation of T cell identity genes and repression of non-T lineage genes, conferring fine control of various T cell functionalities. These findings suggest that Tcf1/Lef1 control global genome organization and help form intricate chromatin-interacting hubs to facilitate promoter-enhancer/silencer contact, hence providing constant supervision of CD8$^+$ T cell identity and function.

SRR_number	datatype	treatment	cell	replicate
SRR13423162	RNAseq	WT	CD8 T cell	1
SRR13423163	RNAseq	WT	CD8 T cell	2
SRR13423164	RNAseq	WT	CD8 T cell	3
SRR13423165	RNAseq	TCF1 - KO	CD8 T cell	1
SRR13423166	RNAseq	TCF1 - KO	CD8 T cell	2
SRR13423167	RNAseq	TCF1 - KO	CD8 T cell	3

SRR_number	datatype	treatment	cell	replicate
SRR13423162	RNAseq	WT	CD8 T cell	1
SRR13423163	RNAseq	WT	CD8 T cell	2
SRR13423164	RNAseq	WT	CD8 T cell	3
SRR13423165	RNAseq	TCF1 - KO	CD8 T cell	1
SRR13423166	RNAseq	TCF1 - KO	CD8 T cell	2
SRR13423167	RNAseq	TCF1 - KO	CD8 T cell	3

This script will contain:

- Variables
- Sed command
- For loop
- Module load
- hisat2
- samtools

Jupyter Notebook version: 04240a1

This app will launch a Jupyter Notebook server on one or more nodes.

Partition

bluemoon

- To request a GPU specify a partition such as dggpu or bdgpu

Number of hours (min-1, max-48)

1

Number of nodes (min-1, max-4)

1

Number of cores per node (min-1, max-32)

1

Number of GPUs per node (min-0, max-2)

0

- If requesting GPU nodes, you must enter a GPU-enabled Partition above or the job will fail.

Or use your favorite text editor!

Extra Modules

- Specify additional environment modules here (space delimited)

How to create a variable

nameofvarible=valueofvarible

To recall the contents of the variable

```
echo $nameofvariable
```

Utility of variables

Variables can be used to store information that can be used later in the script (once or many times over)

Example of Variable usage

```
DBDIR=/users/p/d/pdrodrig/genome_in
dex
```

In this case, this is showing a location to a specific directory

Curly braces {}

- https://devhints.io/bash

- Substitutions, manipulations, etc.. So much more!

backslash \

is used by *bash* to indicate a line continuation and is commonly used in *bash* scripts

Indenting and blank lines

- Indenting is done to clarify your code. Indentations are usually used for loops, if statements, function definitions to make it easy to see what statements are part of that loop or part of the if statement.

- Another trick to make your code more readable is adding blank lines to separate out blocks of related code.

for loop is classified as an iteration statement

For loops are constructed with 4 basic words:

Words	What it does
for	set the loop variable name
in	specify whatever it is we are looping over
do	specify what we want to do with each item
done	tell the computer we are done

for loop syntax

putting these together, for loop syntax is as follows:

```
for VARIABLE in file1 file2 file3
do
  command1 on $VARIABLE
  command2
done
```

A basic loop looks something like this when it's written within a job script:

```
for i in A B C
do
  echo $i
done
```

HISAT2 usage

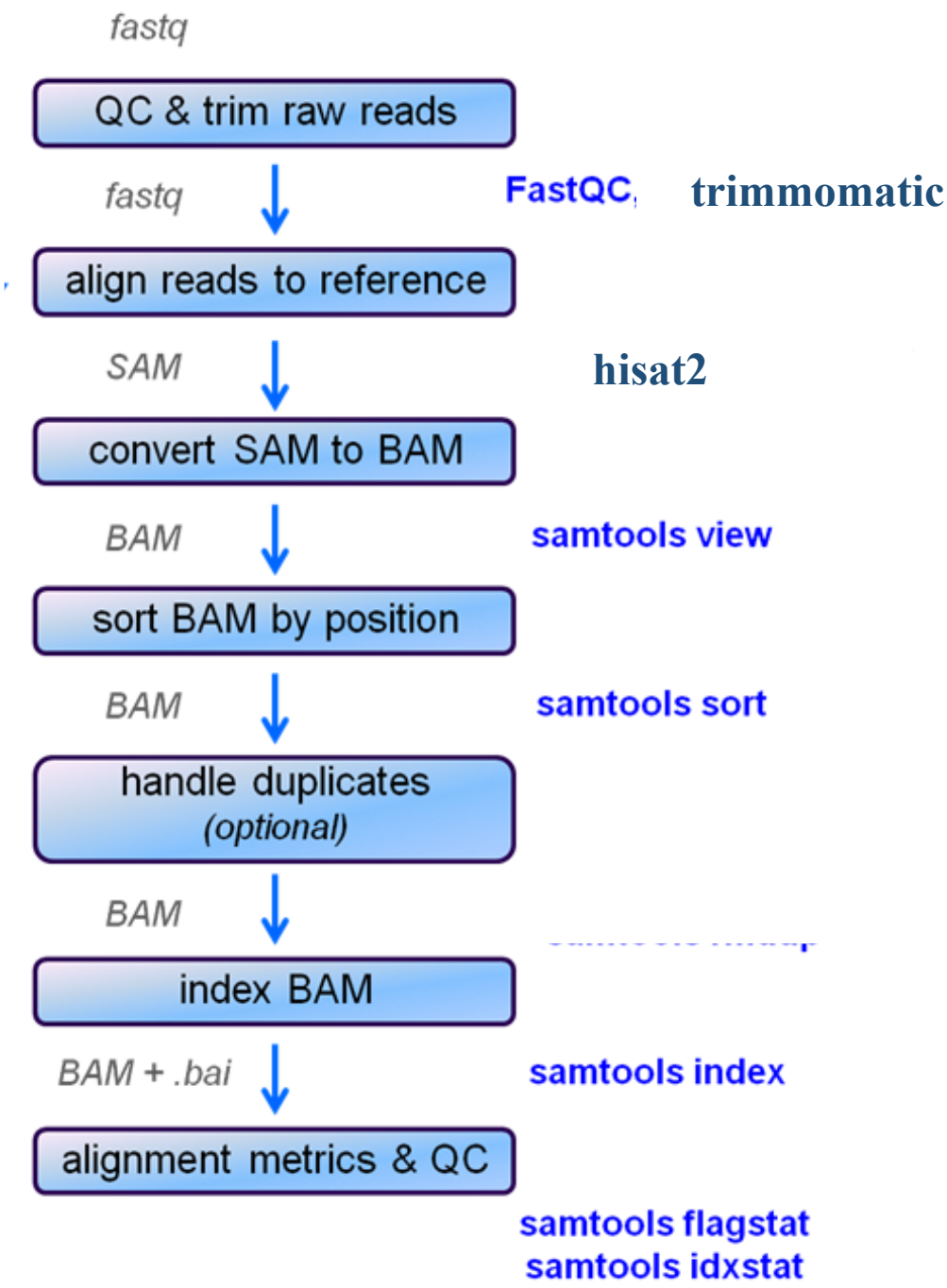
- http://daehwankimlab.github.io/hisat2/

- hisat2 [options]* -x <hisat2-idx> {-1 <m1> -2 <m2> | -U <r> | --sra-acc <SRA accession number>} [-S <hit>]

Notice the naming begins with GRCh38.

```
[pdrodrig@vacc-user1 hisat2_index]$ ls
align_human_7376182.out  GRCh38.3.ht2  GRCh38.8.ht2  GRCm39.5.ht2
align_mouse_7375972.out  GRCh38.4.ht2  GRCm39.1.ht2  GRCm39.6.ht2
align_mouse_7375976.out  GRCh38.5.ht2  GRCm39.2.ht2  GRCm39.7.ht2
GRCh38.1.ht2             GRCh38.6.ht2  GRCm39.3.ht2  GRCm39.8.ht2
GRCh38.2.ht2             GRCh38.7.ht2  GRCm39.4.ht2  hisat2_index_script
```

Alignment Workflow



hisat2_align.sh

This will not be "ready-to-go" for everyone, you may need to add upon it for your dataset

Basic Template

```
#!/bin/bash
#SBATCH --partition=bluemoon
#SBATCH --nodes=1
#SBATCH --ntasks=2
#SBATCH --mem=10G
#SBATCH --time=3:00:00
#SBATCH --job-name=align_CD8
# %x=job-name %j=jobid
#SBATCH --output=%x_%j.out

for i in *fastq.gz; do
SAMPLE=$(echo ${i} | sed "s/.fastq.gz//")
echo ${SAMPLE}.fastq.gz

DBDIR=/gpfs1/cl/mmg232/course_materials/hisat2_index
GENOME="GRCm39"
p=2

module load hisat2-2.1.0-gcc-7.3.0-knvgwpc
module load samtools-1.10-gcc-7.3.0-pdbkohx

#align to GRCm39
hisat2 \
  -p ${p} \
  -x ${DBDIR}/${GENOME} \
  -U ${SAMPLE}.fastq.gz \
  -S ${SAMPLE}.sam &> ${SAMPLE}.log

#create bam file
samtools view ${SAMPLE}.sam \
  --threads 2 \
  -b \
  -o ${SAMPLE}.bam \
```

1

2

3

4

Read the methods

- Did the authors add special arguments during alignment?
- **TRY TO** understand why this was done.
- Email me and we can chat!

After running HISAT2 your outputs will look like this:

After running the hisat2_align.sh script with the FASTQ provided, the outputs will look like this:

```
align_CD8_7422840.out  SRR13423162.log              SRR13423165.fastq.gz
hisat2_align.sh        SRR13423162_sorted.bam       SRR13423165.log
hisat2.log             SRR13423162_sorted.bam.bai   SRR13423165_sorted.bam
SRR13423162.bam        SRR13423162.txt              SRR13423165_sorted.bam.bai
SRR13423162.fastq.gz   SRR13423165.bam              SRR13423165.txt
```

*.bam
*.log
*.bai
*.txt

SAMtools usage

- http://www.htslib.org/doc/samtools.html

samtools view [*options*] *in.sam*|*in.bam*|*in.cram* [*region*...]

samtools sort [-l *level*] [-u] [-m *maxMem*] [-o *out.bam*] [-O *format*] [-M] [-K *kmerLen*] [-n] [-t *tag*] [-T *tmpprefix*] [-@ *threads*] [*in.sam*|*in.bam*|*in.cram*]

samtools index [-bc] [-m *INT*] *aln.sam*|*aln.bam*|*aln.cram* [*out.index*]

Read alignments files: the SAM format

- 'Sequence Alignment/Map' format - http://samtools.sourceforge.net/SAMv1.pdf
- SAM/BAM files can be manipulated with SAMtools
- SAM files are tab-delimited files, human-readable
- The SAM file contains two sections:

1. Header section:
   - Metadata about the genome, the samples, the pipeline
   - Header lines start with @
2. Alignments (or 'records') section

To view a SAM file:

module load samtools-1.10-gcc-7.3.0-pdbkohx

samtools view -h SRR13423162_sorted.bam | less -S

SAM header descriptions

Tag		Description
@HD		The header line. The first line if present.
	VN*	Format version. *Accepted format*: /^[0-9]+\.[0-9]+$/.
	SO	Sorting order of alignments. *Valid values*: **unknown** (default), **unsorted**, **queryname** and **coordinate**. For coordinate sort, the major sort key is the RNAME field, with order defined by the order of @SQ lines in the header. The minor sort key is the POS field. For alignments with equal RNAME and POS, order is arbitrary. All alignments with '*' in RNAME field follow alignments with some other value but otherwise are in arbitrary order.
	GO	Grouping of alignments, indicating that similar alignment records are grouped together but the file is not necessarily sorted overall. *Valid values*: **none** (default), **query** (alignments are grouped by QNAME), and **reference** (alignments are grouped by RNAME/POS).
@SQ		Reference sequence dictionary. The order of @SQ lines defines the alignment sorting order.
	SN*	Reference sequence name. The SN tags and all individual AN names in all @SQ lines must be distinct. The value of this field is used in the alignment records in RNAME and RNEXT fields. Regular expression: [!-)+-<>-~][!-~]*
	LN*	Reference sequence length. *Range*: [1,2^{31}-1]
	AH	Indicates that this sequence is an alternate locus.[4] The value is the locus in the primary assembly for which this sequence is an alternative, in the format '*chr*:*start*-*end*', '*chr*' (if known), or '*' (if unknown), where '*chr*' is a sequence in the primary assembly. Must not be present on sequences in the primary assembly.
	AN	Alternative reference sequence names. A comma-separated list of alternative names that tools may use when referring to this reference sequence.[5] These alternative names are not used elsewhere within the SAM file; in particular, they must not appear in alignment records' RNAME or RNEXT fields. *Regular expression*: *name*(,*name*)* where *name* is [0-9A-Za-z][0-9A-Za-z*+.@_|-]*
	AS	Genome assembly identifier.
	M5	MD5 checksum of the sequence. See Section 1.3.1
	SP	Species.
	UR	URI of the sequence. This value may start with one of the standard protocols, e.g http: or ftp:. If it does not start with one of these protocols, it is assumed to be a file-system path.
@RG		Read group. Unordered multiple @RG lines are allowed.
	ID*	Read group identifier. Each @RG line must have a unique ID. The value of ID is used in the RG tags of alignment records. Must be unique among all read groups in header section. Read group IDs may be modified when merging SAM files in order to handle collisions.
	CN	Name of sequencing center producing the read.
	DS	Description.
	DT	Date the run was produced (ISO8601 date or date/time).
	FO	Flow order. The array of nucleotide bases that correspond to the nucleotides used for each flow of each read. Multi-base flows are encoded in IUPAC format, and non-nucleotide flows by various other characters. *Format*: /*|[ACMGRSVTWYHKDBN]+/

SAM alignment section

cf. FASTQ format

Read Name | FLAG | Chrom | AlnStart | CIGAR | Sequence | BaseQuals

```
6_1303_10584_85775 99    groupVIII  311  3   63M3I34M  =  780  572  GGGTATTGGGC  @CFFFFFHFH
6_1111_20943_90813 163   groupVIII  315  40  100M      =  809  594  TAATGAAGCCAT  @BDDFDDA+<A<
6_2111_2016_88235  355   groupVIII  315  3   100M      =  856  573  TAATGAAGCCAT  @?DADDBD>D>B
6_1104_8139_99999  163   groupVIII  316  14  100M      =  818  602  AATGAAGCCATT  @@FFFFFGHGHH
6_1304_4167_91751  163   groupVIII  322  5   52M3I29M  =  812  573  GCCATTTTTAC   <<BDBDEHHDF
6_2301_14383_16382 163   groupVIII  323  40  51M3I46M  =  809  589  CCATTTTTACT   CCFFFFFHHHH
```

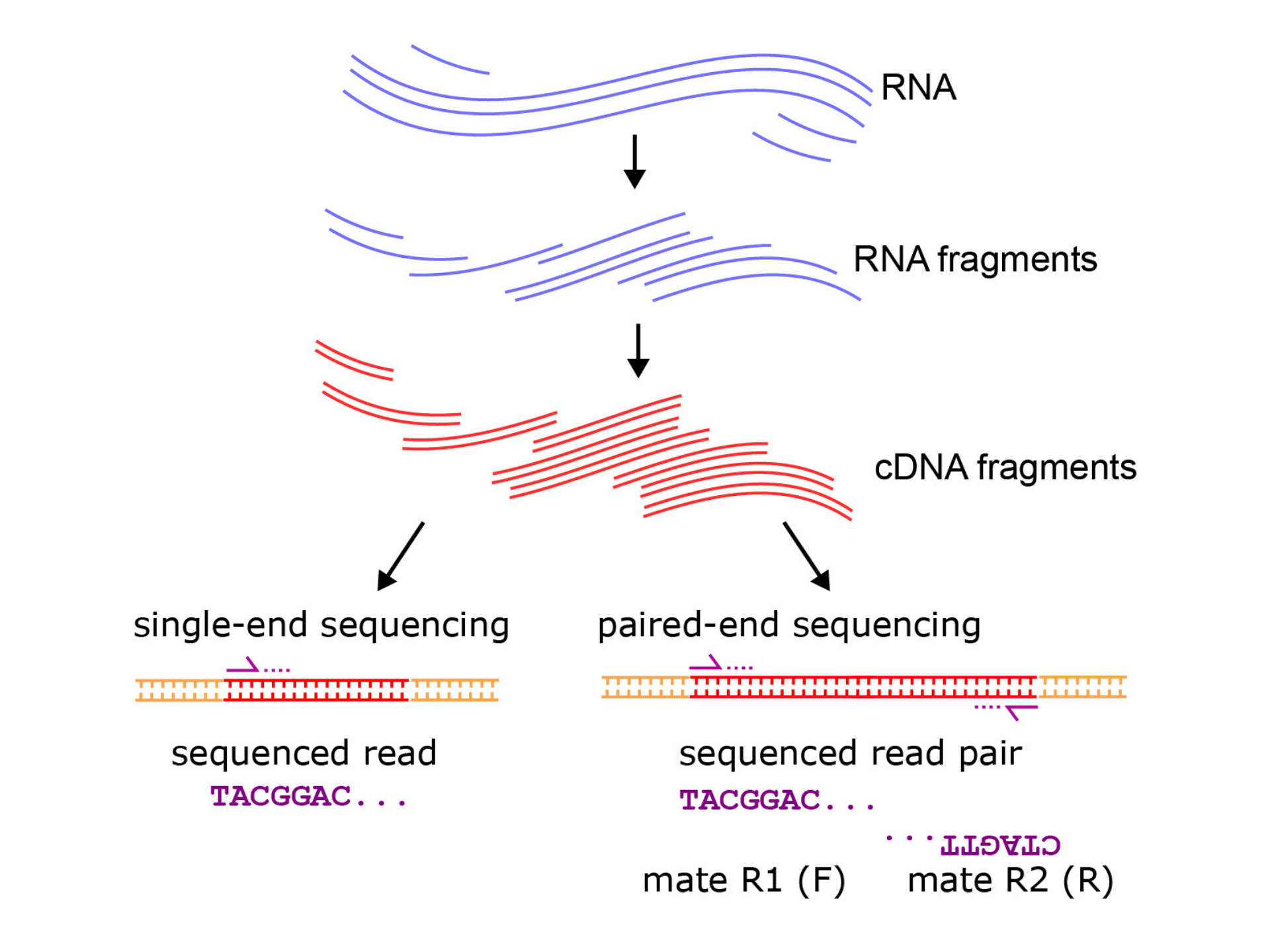
Col	Field	Type	Regexp/Range	Brief description
1	QNAME	String	`[!-?A-~]{1,255}`	Query template NAME
2	FLAG	Int	$[0,2^{16}-1]$	bitwise FLAG
3	RNAME	String	`\*|[!-()+-<>-~][!-~]*`	Reference sequence NAME
4	POS	Int	$[0,2^{31}-1]$	1-based leftmost mapping POSition
5	MAPQ	Int	$[0,2^{8}-1]$	MAPping Quality
6	CIGAR	String	`\*|([0-9]+[MIDNSHPX=])+`	CIGAR string
7	RNEXT	String	`\*|=|[!-()+-<>-~][!-~]*`	Ref. name of the mate/next read
8	PNEXT	Int	$[0,2^{29}-1]$	Position of the mate/next read
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

Column 2: Bitwise Flag

- Is a lookup code to explain certain features about the particular read
- It tells you whether the read aligned, is marked as a PCR duplicate, if its mate aligned, etc.

Bit		Description
1	0x1	template having multiple segments in sequencing
2	0x2	each segment properly aligned according to the aligner
4	0x4	segment unmapped
8	0x8	next segment in the template unmapped
16	0x10	SEQ being reverse complemented
32	0x20	SEQ of the next segment in the template being reverse complemented
64	0x40	the first segment in the template
128	0x80	the last segment in the template
256	0x100	secondary alignment
512	0x200	not passing filters, such as platform/vendor quality controls
1024	0x400	PCR or optical duplicate
2048	0x800	supplementary alignment

A combination of the flags, results in one integer, which makes it difficult to interpret

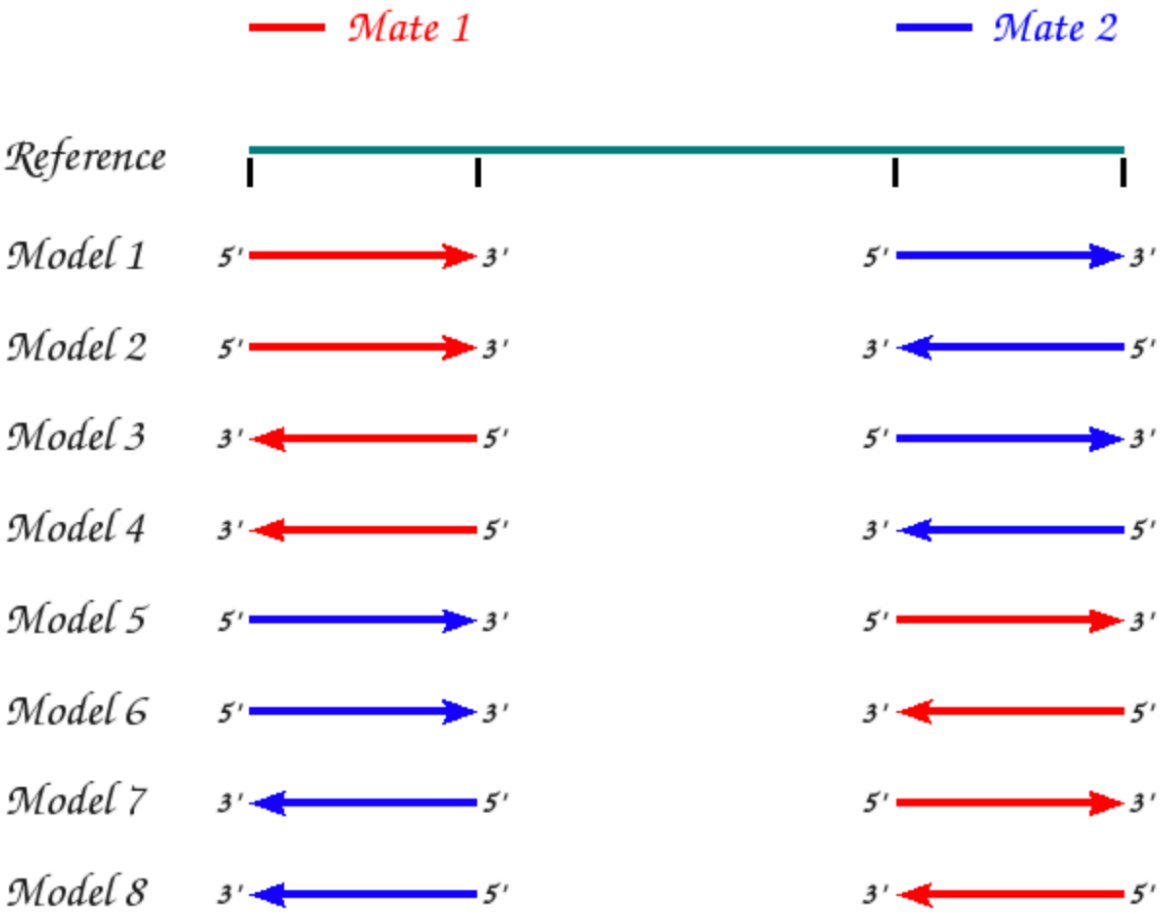

RNA
RNA fragments
cDNA fragments
single-end sequencing
paired-end sequencing
sequenced read
TACGGAC...
sequenced read pair
TACGGAC...
...CTAGTT
mate R1 (F)
mate R2 (R)

“Proper” mate-pairing

Same direction

Face-in

Face-out



A combination of the flags, results in one integer, which makes it difficult to interpret

https://broadinstitute.github.io/picard/explain-flags.html

Column 6: 'CIGAR strings'

6_1303_10584_85775	99	groupVIII	311	3	63M3I34M
6_1111_20943_90813	163	groupVIII	315	40	100M
6_2111_2016_88235	355	groupVIII	315	3	100M
6_1104_8139_99999	163	groupVIII	316	14	100M
6_1304_4167_91751	163	groupVIII	322	5	52M3I29M
6_2301_14383_16382	163	groupVIII	323	40	51M3I46M

100M — 100 matching nucleotides (i.e. no gaps)

63M-3I-34M — 63 matching nucleotides
3 nucleotides not in the reference (3bp insertion)
34 matching nucleotides

Aligned Read

TGCAGGATGGATGTGTTCCTCCTCAGCTGCTTATTTTAACTCCACTGCACAACATGTTTTGTGTTATATTCTTTCGCTGTGTAGTCTGTAAGC

TGCAGGGACTGCAGGATGGATGTGTTCCTCCTCAGCTGCTTATTTTAACTCCAC---ACAACATGTTTTGTGTTATATTCTTTCGCTGTGTAGTCTGTAAGCAGAGTATGATACTG

Reference

Column 6: 'CIGAR strings'

Op	BAM	Description
M	0	alignment match (can be a sequence match or mismatch)
I	1	insertion to the reference
D	2	deletion from the reference
N	3	skipped region from the reference
S	4	soft clipping (clipped sequences present in SEQ)
H	5	hard clipping (clipped sequences NOT present in SEQ)
P	6	padding (silent deletion from padded reference)
=	7	sequence match
X	8	sequence mismatch

- Example: intron = 81 bases

ERR022486.8388510 81 22 32099 255 **58M81N18M** = 27484 -4772
CCTTGGTCTTGCCGAAGTAGATCTCATTGAGAGTGGAGCGGATCTTGTTCTCCATTTCCTCCA
CCAGGCGTCCGAT :9=<==;<<><=><?>>?<?==>>?>><?>>??<AA?
@AFADDD;GDGAG@GGCBE@GG?GG>GGGG?GGGGGGGG NM:i:0 XS:A:- NH:i:1

Aligned Read 58M 81N 18M ***Spliced***

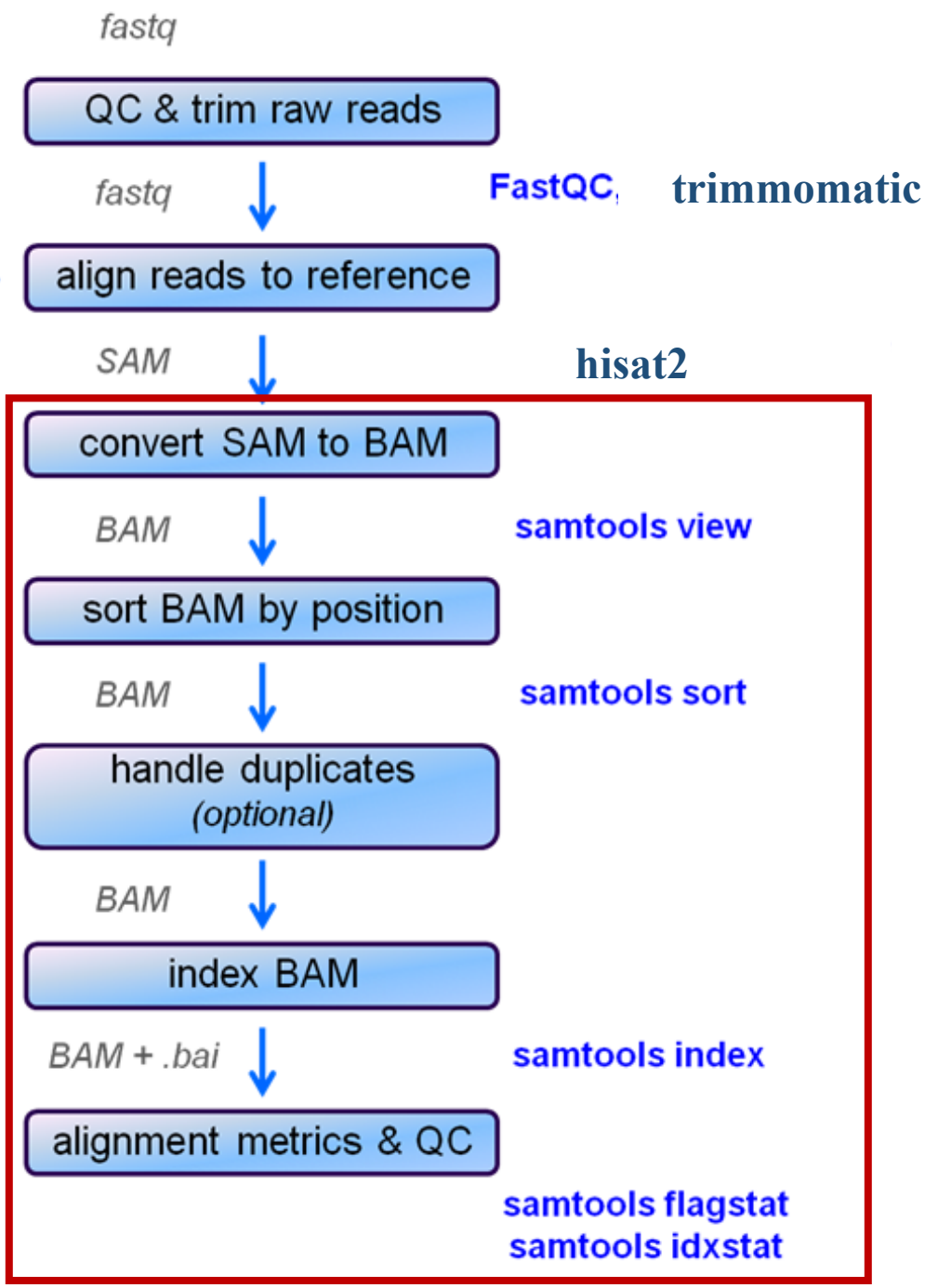
TGCAGGATGGATGTGTTCCTCCTCAGCTGCTTA----------TATATTCTTTCGCTGTGTAGTCTGTAAGC

TGCAGGGACTGCAGGATGGATGTGTTCCTCCTCAGCTGCTTATTTTAACTCCACACAACATGTTTTGTGTTATATTCTTTCGCTGTGTAGTCTGTAAGCAGAGTATGATACTG

Reference

We never want to keep a SAM file - so in our workflow we immediately convert it to a BAM file

Alignment Workflow



BAM file

- BAM (Binary Alignment/Map) format:

- Compressed binary representation of SAM
- Greatly reduces storage space requirements to about ~27% of original SAM
- Not human-readable

Common order of operations

1. SAM files are converted to BAM files (*samtools view*)
2. BAM files are sorted by reference coordinates (*samtools sort*)
3. SORTED BAM files are indexed (*samtools index*)

samtools view

```
samtools view -b input.sam > input.bam
```

- Input is usually a SAM file, but can also use a BAM
- Common uses: extracting a subset of data into a new file, converting between SAM/BAM files, or just viewing raw files

samtools sort

```
samtools sort sample.bam -o sample.sorted.bam
```

- Reads need to be ordered in “genomic order” – not the order in which they were sequenced

samtools index

```
samtools index sorted.bam
```

- Creates index file that allows for fast look-up
- Generates *.bam.bai file

Why create a bam file?

- Show UCSC Custom tracks
- TLR3 vs TLR2

Outline

- Class Activity #1, ~5 minutes
- Lecture for ~25 minutes
- **Class Activity #2, ~15 minutes**