Lecture 13

RNA-seq: Alignment

STAR

BIOINFORMATICS ORIGINAL PAPER

Vol. 29 no. 1 2013, pages 15–21
doi:10.1093/bioinformatics/bts635

Sequence analysis

Advance Access publication October 25, 2012

STAR: ultrafast universal RNA-seq aligner

Alexander Dobin[1,*], Carrie A. Davis[1], Felix Schlesinger[1], Jorg Drenkow[1], Chris Zaleski[1], Sonali Jha[1], Philippe Batut[1], Mark Chaisson[2] and Thomas R. Gingeras[1]

[1]Cold Spring Harbor Laboratory, Cold Spring Harbor, NY, USA and [2]Pacific Biosciences, Menlo Park, CA, USA

Associate Editor: Inanc Birol

Benchmarking of RNA-seq aligners

Simulation-based comprehensive benchmarking of RNA-seq aligners

Giacomo Baruzzo[1,5], Katharina E Hayer[2,5], Eun Ji Kim[2], Barbara Di Camillo[1], Garret A FitzGerald[2,3] & Gregory R Grant[2,4]

Alignment is the first step in most RNA-seq analysis pipelines, and the accuracy of downstream analyses depends heavily on it. Unlike most steps in the pipeline, alignment is particularly amenable to benchmarking with simulated data. We performed a comprehensive benchmarking of 14 common splice-aware aligners for base, read, and exon junction-level accuracy and compared default with optimized parameters. We found that performance varied by genome complexity, and accuracy and popularity were poorly correlated. The most widely cited tool underperforms for most metrics, particularly when using default settings.

Simulating data for benchmarking alignment algorithms is straightforward on account of the discrete nature of the data. Simulated data were used for comprehensive RNA-seq alignment benchmarking studies in 2011 (ref. 8) and 2013 (ref. 13), but alignment methods have undergone considerable development since then. Here we analyze performance at the base, read, and junction levels using default and optimized parameters. We also examine execution time and memory usage; differential behavior at canonical versus noncanonical junctions; the effect of untrimmed adapters; performance on indels, reads that map to multiple sites (multimappers); and other factors.

Benchmarking of RNA-seq aligners

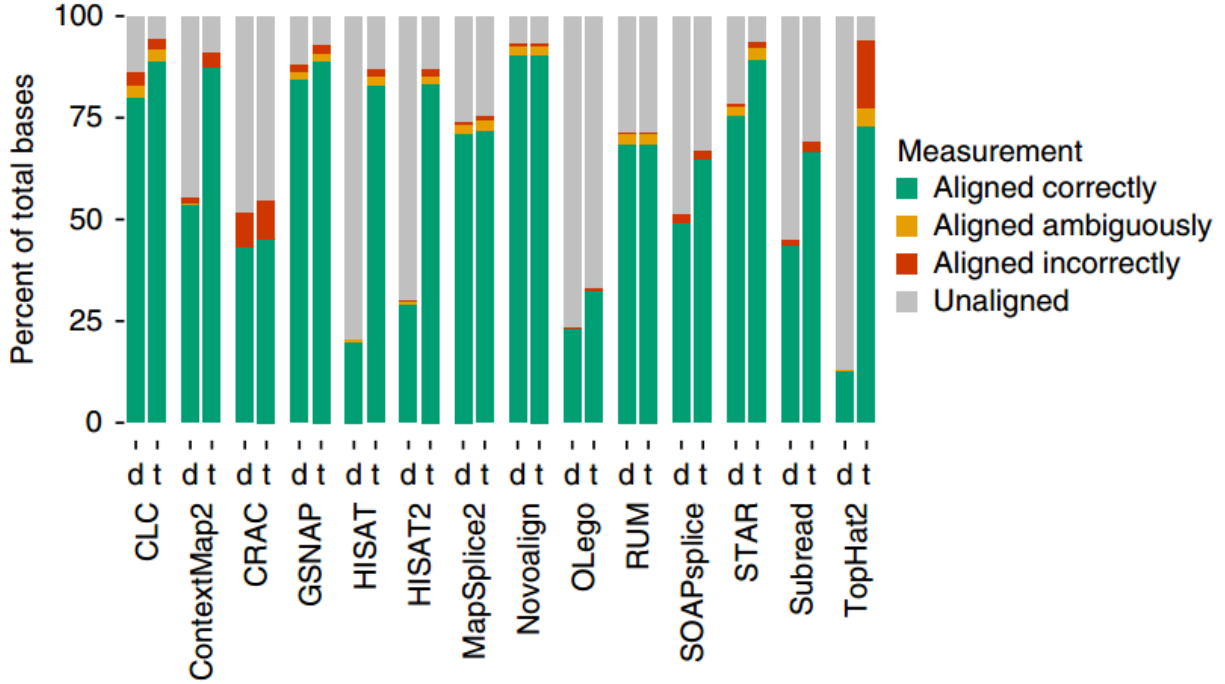
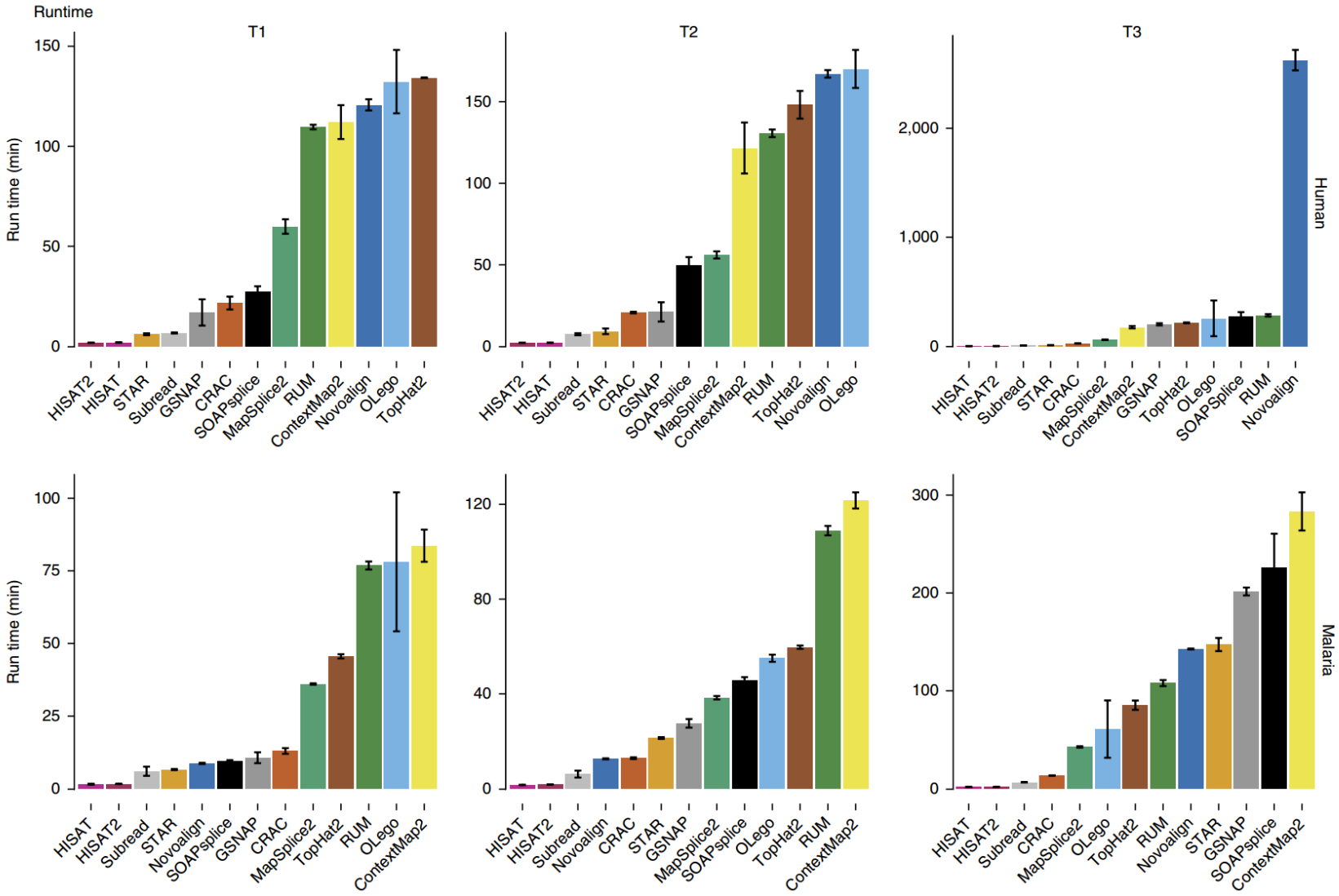


Figure 3 | The effect of tuning parameters on the human-T3-data base-level statistics. For each tool, the figure shows the alignment statistics for the 'default' (d) and the 'tuned' (t) alignments.

Benchmarking of RNA-seq aligners



Benchmarking of RNA-seq aligners

“Based on this analysis the most reliable general-purpose aligners appear to be CLC, Novoalign, GSNAP, and STAR.”

STAR alignment strategy

STAR consists of a two-step process:

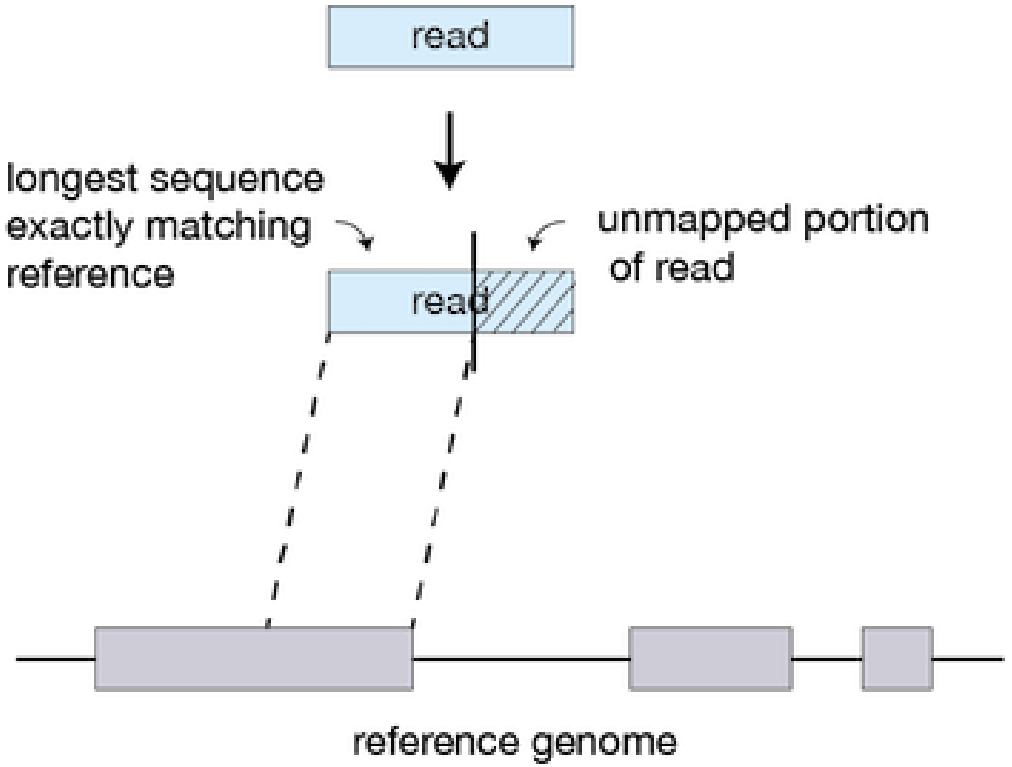
1. Seed searching: sequential searching of only the unmapped portions of reads.

2. Clustering, stitching, and scoring: building alignments of the entire read sequence by stitching together all the seeds that were aligned to the genome.

Seed searching

For each read, STAR searches for the longest sequence that exactly matches one or more locations on the reference genome. This longest matching sequence is called the MMP (seed1).

Maximal mappable prefix (MMP)

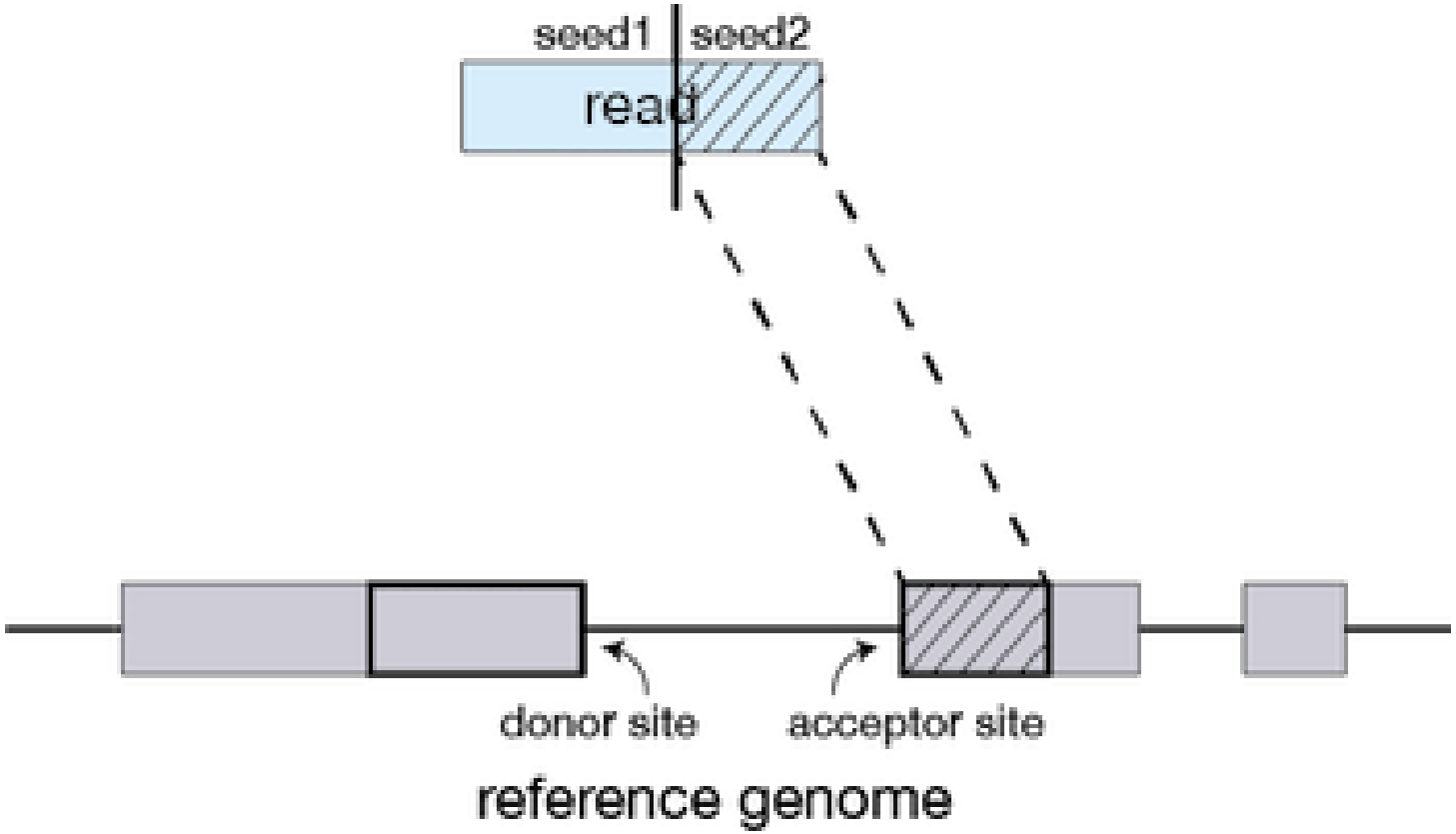


STAR uses an uncompressed suffix array to efficiently search for the MMPs.

https://hbctraining.github.io/Intro-to-rnaseq-hpc-O2/lessons/03_alignment.html

Seed searching

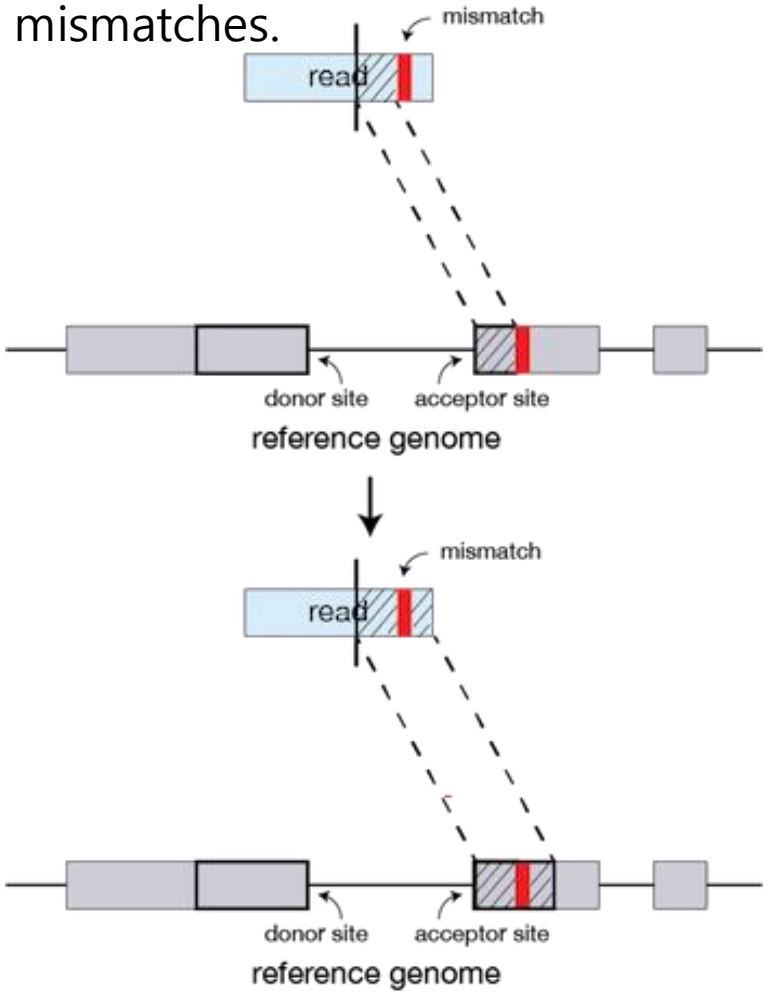
STAR will search again for only the unmapped portion of the read to find the next MMP (seed2)



https://hbctraining.github.io/Intro-to-rnaseq-hpc-O2/lessons/03_alignment.html

Seed searching

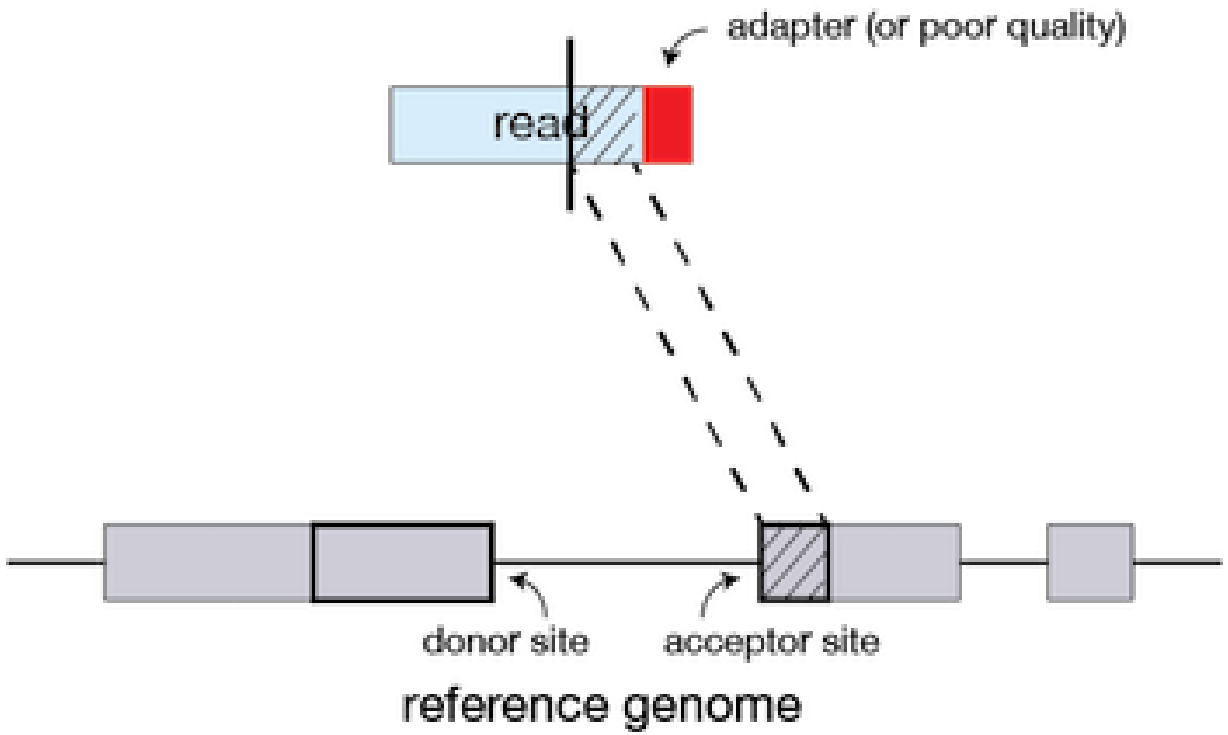
If the MMP search does not reach the end of a read due to mismatches or indels, the previous MMPs will be extended, allowing for alignments with mismatches.



https://hbctraining.github.io/Intro-to-rnaseq-hpc-O2/lessons/03_alignment.html

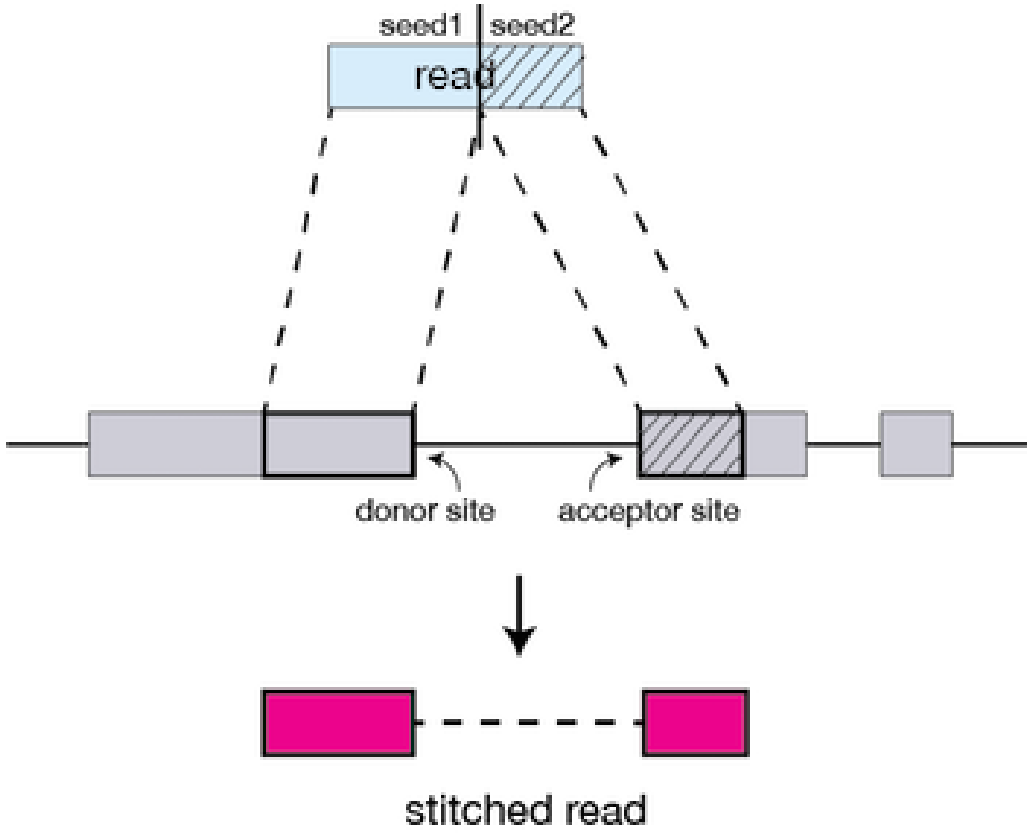
Seed searching

If extension does not find a good alignment, then the poor quality or adaptor sequence will be soft clipped.



https://hbctraining.github.io/Intro-to-rnaseq-hpc-O2/lessons/03_alignment.html

Clustering, stitching, and scoring



https://hbctraining.github.io/Intro-to-rnaseq-hpc-O2/lessons/03_alignment.html

Clustering, stitching, and scoring

First, the seeds are clustered together by proximity to a selected set of “anchor” seeds.

All the seeds that map within user-defined genomic windows around the anchors are stitched together. The size of the genomic windows determines the maximum intron size for the spliced alignments.

A dynamic programming algorithm is used to stitch each pair of seeds, allowing for any number of mismatches but only one insertion or deletion.

Clustering, stitching, and scoring

The seeds from the matches of paired-end RNA-seq reads are clustered and stitched together.

The stitching is guided by a local alignment scoring scheme, with user-defined scores for matches, mismatches, insertions, deletions and splice junction gaps, allowing for a quantitative assessment of the alignment qualities and ranks.

The stitched combination with the highest score is chosen as the best alignment of a read. For multi-mapping reads, all alignments with scores within a certain user-defined range below the highest score are reported.

Clustering, stitching, and scoring

If an alignment within one genomic window does not cover the entire read sequence, STAR will try to find two or more windows that cover the entire read, resulting in a chimeric alignment, with different parts of the read mapping to distal genomic loci, or different chromosomes, or different strands.

Generating genome indexes

```
--runThreadN NumberOfThreads
--runMode genomeGenerate
--genomeDir /path/to/genomeDir
--genomeFastaFiles /path/to/genome/fasta1 /path/to/genome/fasta2 ...
--sjdbGTFfile /path/to/annotations.gtf
--sjdbOverhang ReadLength-1
```

`--runThreadN` option defines the number of threads to be used for genome generation, it has to be set to the number of available cores on the server node.

`--runMode genomeGenerate` option directs STAR to run genome indices generation job.

`--genomeDir` specifies path to the directory (henceforth called "genome directory" where the genome indices are stored. This directory has to be created (with `mkdir`) before STAR run and needs to have writing permissions. The file system needs to have at least 100GB of disk space available for a typical mammalian genome. It is recommended to remove all files from the genome directory before running the genome generation step. This directory path will have to be supplied at the mapping step to identify the reference genome.

Generating genome indexes

`--genomeFastaFiles` specified one or more FASTA files with the genome reference sequences. Multiple reference sequences (henceforth called chromosomes) are allowed for each fasta file. You can rename the chromosomes names in the chrName.txt keeping the order of the chromosomes in the file: the names from this file will be used in all output alignment files (such as .sam). The tabs are not allowed in chromosomes names, and spaces are not recommended.

Generating genome indexes

`--sjdbGTFfile` specifies the path to the file with annotated transcripts in the standard GTF format. STAR will extract splice junctions from this file and use them to greatly improve accuracy of the mapping. While this is optional, and STAR can be run without annotations, using annotations is **highly recommended** whenever they are available. Starting from 2.4.1a, the annotations can also be included on the fly at the mapping step.

`--sjdbOverhang` specifies the length of the genomic sequence around the annotated junction to be used in constructing the splice junctions database. Ideally, this length should be equal to the *ReadLength-1*, where *ReadLength* is the length of the reads. For instance, for Illumina 2x100b paired-end reads, the ideal value is 100-1=99. In case of reads of varying length, the ideal value is *max(ReadLength)-1*. **In most cases, the default value of 100 will work as well as the ideal value.**

Generating genome indexes

2.2.1 Which chromosomes/scaffolds/patches to include?

It is strongly recommended to include major chromosomes (e.g., for human chr1-22,chrX,chrY,chrM,) as well as un-placed and un-localized scaffolds. Typically, un-placed/un-localized scaffolds add just a few MegaBases to the genome length, however, a substantial number of reads may map to ribosomal RNA (rRNA) repeats on these scaffolds. These reads would be reported as unmapped if the scaffolds are not included in the genome, or, even worse, may be aligned to wrong loci on the chromosomes. Generally, patches and alternative haplotypes should **not** be included in the genome.

Examples of acceptable genome sequence files:

- **ENSEMBL:** files marked with .dna.primary.assembly, such as: ftp://ftp.ensembl.org/pub/release-77/fasta/homo_sapiens/dna/Homo_sapiens.GRCh38.dna.primary_assembly.fa.gz
- **NCBI:** "no alternative - analysis set": ftp://ftp.ncbi.nlm.nih.gov/genbank/genomes/Eukaryotes/vertebrates_mammals/Homo_sapiens/GRCh38/seqs_for_alignment_pipelines/GCA_000001405.15_GRCh38_no_alt_analysis_set.fna.gz

Generating genome indexes

2.2.2 Which annotations to use?

The use of the most comprehensive annotations for a given species is strongly recommended. Very importantly, chromosome names in the annotations GTF file have to match chromosome names in the FASTA genome sequence files. For example, one can use ENSEMBL FASTA files with ENSEMBL GTF files, and UCSC FASTA files with UCSC FASTA files. However, since UCSC uses `chr1, chr2, ...` naming convention, and ENSEMBL uses `1, 2, ...` naming, the ENSEMBL and UCSC FASTA and GTF files cannot be mixed together, unless chromosomes are renamed to match between the FASTA anf GTF files.

For mouse and human, the Gencode annotations are recommended: http://www.gencodegenes.org/.

Generating genome indexes

```
(base) cat all.sh
#!/bin/bash
set -e
set -u
set -o pipefail

wget -P ~/reference/human ftp://ftp.ensembl.org/pub/release-108/fasta/homo_sapiens/dna/Homo_sapiens.GRCh38.dna.primary_assembly.fa.gz
wget -P ~/reference/human ftp://ftp.ensembl.org/pub/release-108/gtf/homo_sapiens/Homo_sapiens.GRCh38.108.gtf.gz
```

GTF and FASTA files are downloaded in

`/home/reference/human/Homo_sapiens.GRCh38.108.gtf.gz`

`/home/reference/human/Homo_sapiens.GRCh38.dna.primary_assembly.fa.gz`

Generating genome indexes

```
(base) sum /home/reference/human/Homo_sapiens.GRCh38.108.gtf.gz
31848 52840
(base) sum /home/reference/human/Homo_sapiens.GRCh38.dna.primary_assembly.fa.gz
02415 860559
```

```
31848 52840 Homo_sapiens.GRCh38.108.gtf.gz

02415 860559 Homo_sapiens.GRCh38.dna.primary_assembly.fa.gz
```

Generating genome indexes

```
gunzip /home/reference/human/Homo_sapiens.GRCh38
.dna.primary_assembly.fa.gz
gunzip /home/reference/human/Homo_sapiens.GRCh38
.108.gtf.gz
```

Generating genome indexes

```
STAR --runThreadN 10 --runMode genomeGenerate --genomeDir
/home/username/reference/human/star \

--genomeFastaFiles /home/reference/human/Homo_sapiens.GRC
h38.dna.primary_assembly.fa \

--sjdbGTFfile /home/reference/human/Homo_sapiens.GRCh38.1
08.gtf --sjdbOverhang 100
```

Generating genome indexes

```
(base) cat star_genome.sh
#!/bin/bash
set -e
set -u
set -o pipefail

STAR --runThreadN 10 --runMode genomeGenerate --genomeDir /home/yjjeong/reference/human/star \
        --genomeFastaFiles /home/reference/human/Homo_sapiens.GRCh38.dna.primary_assembly.fa \
        --sjdbGTFfile /home/reference/human/Homo_sapiens.GRCh38.108.gtf --sjdbOverhang 100
```

Generating genome indexes

```
sh star_genome.sh
```

Example: Running mapping jobs

```
sample_PE='SRR3191545'

STAR --runThreadN 10 --runMode alignReads \

--genomeDir /home/username/reference/human/star --readFilesIn
/home/Data/zika/${sample_PE}_1.fastq.gz
/home/Data/zika/${sample_PE}_2.fastq.gz --readFilesCommand zcat \

--outFileNamePrefix /home/username/star/${sample_PE}_ --
outFilterMultimapNmax 1 --outSAMtype BAM Unsorted \

--outSAMattributes All --quantMode GeneCounts --sjdbGTFfile
/home/reference/human/Homo_sapiens.GRCh38.108.gtf \

--twopassMode Basic
```

Example: Running mapping jobs

```
(base) cat run_star_pe.example.sh
#!/bin/bash
set -e
set -u
set -o pipefail

sample_PE='SRR3191545'

STAR --runThreadN 10 --runMode alignReads \
        --genomeDir /home/yjjeong/reference/human/star \
        --readFilesIn /home/Data/zika/${sample_PE}_1.fastq.gz /home/Data/zika/${sample_PE}_2.fastq.gz \
        --readFilesCommand zcat \
        --outFileNamePrefix /home/yjjeong/star/${sample_PE}_ \
        --outFilterMultimapNmax 1 --outSAMtype BAM Unsorted \
        --outSAMattributes All --quantMode GeneCounts \
        --sjdbGTFfile /home/reference/human/Homo_sapiens.GRCh38.108.gtf \
        --twopassMode Basic
```

Basic options for running mapping jobs

`--runThreadN`: Number of treads

`--genomeDir`: /path/to/genomeDir

`--readFilesIn`: /path/to/read1 [/path/to/read2]

`--readFilesIn`

- Names (with path) of the files containing the sequences to be mapped.

- If the read files are compressed, use the `--readFilesCommand` option. For gzipped files, use
  `--readFilesCommand zcat` or
  `--readFilesCommand gunzip -c`

`--readFilesIn`

- SE: `--readFilesIn read.fastq`
- PE: `--readFilesIn read1.fastq read2.fastq`
- Multiple SE files: `--readFilesIn A.fastq,B.fastq,C.fastq`

Spaces are not allowed in the comma-separated list.

- Multiple PE files: `--readFilesIn A1.fastq,B1.fastq,C1.fastq A2.fastq, B2.fastq,C2.fastq`

Space separates the comma-separated lists for the read1 and read2.

Using annotations at the mapping stage

- Annotations can be included on the fly at the mapping step, without including them at the genome generation step.

- You can specify

```
--sjdbGTFfile /path/to/ann.gtf
--sjdbOverhang
```

- You can skip this if provided during database creation step.

Using annotations at the mapping stage

- The junctions in these files will be added on-the-fly to the junctions that were included at the genome generation step.

- If `--sjdbOverhang` parameter is supplied at both genome generation and mapping steps, they have to match.

- If `--sjdbOverhang` parameter is not set at the mapping step, it will be set to the one supplied at the genome generation step.

Output files

- STAR produces multiple output files (`Log.XXX`, `Aligned.out.sam`, `SJ.out.tab`).

- All files have standard name, however, you can change the file prefix (including full or relative path) using

`--outFileNamePrefix /path/to/output/dir/prefix`

- By default, this parameter is `./`, i.e. all output files are written in the current directory.

Log files

- Log.out: main log file with a lot of detailed information about the run. This file is most useful for troubleshooting and debugging.

- Log.progress.out: reports job progress statistics, such as the number of processed reads, % of mapped reads etc. It is updated in 1 minute intervals.

- Log.final.out: summary mapping statistics after mapping job is complete, very useful for quality control. The statistics are calculated for each read (single- or paired-end) and then summed or averaged over all reads. Note that STAR counts a paired-end read as one read. Most of the information is collected about the UNIQUE mappers.

SAM/BAM

- By default, the alignments are output in the SAM format into the Aligned.out.sam.

- The mate alignments for PE are adjacent to each other. All multimapping alignments are output consecutively.

- STAR can output alignments directly in BAM format. It can also sort BAM files by coordinates, which is required by many downstream applications.

SAM/BAM: Sorted

`--outSAMtype SAM/BAM/None [Unsorted/SortedByCoordinate]`

- `--outSAMtype BAM Unsorted`: output unsorted `Aligned.out.bam` file. The paired ends of an alignment are always adjacent, and multiple alignments of a read are adjacent as well. This “unsorted” file can be directly used with downstream software such as HTSeq, without the need of name sorting.

SAM/BAM: Sorted

- `--outSAMtype BAM SortedByCoordinate`: output sorted by coordinate `Aligned.sortedByCoord.out.bam` file, similar to samtools sort command.

- `--outSAMtype BAM Unsorted SortedByCoordinate`: output both unsorted and sorted files.

SAM/BAM: Unmapped reads

- By default, STAR does not output unmapped reads. The unmapped reads can be output within the SAM/BAM files using the `--outSAMunmapped Within` option.

- This creates a complete SAM/BAM file containing information about all input reads, which allows recreation of the original FASTQ file with the exception of read order.

- Another option, `--outReadsUnmapped Fastx`, allows output of unmapped reads into separate files, FASTA or FASTQ.

SAM/BAM: Attributes

STAR can output several SAM attributes, which are controlled by the following option:

`--outSAMattributes <String>`

default: `Standard`

string: a string of desired SAM attributes, in the order desired for the output SAM

`NH`
: any combination in any order

`Standard`
: NH HI AS nM

`All`
: NH HI AS nM NM MD jM jI ch

`None`
: no attributes

SAM/BAM: Attributes

- NH: number of loci a read maps to (1 for unique mappers, >1 for multimappers)
- HI: hit index for the multimapping alignments
- AS: alignment score
- nM: number of mismatches (sum from both mates for paired-end reads)
- NM: edit distance (number of mismatches + number of insertion/ deletion bases) for each mate
- MD: string for mismatching positions

SAM/BAM: Read groups

- Read groups can be added to SAM/BAM records while mapping using --outSAMattrRGline.

- The read group fields should be separated by space, and the first field should be “ID:<rg-id>”

Standard output

- Some of the output files can be redirected into the standard output, which may facilitate in creating the pipelines:

--outStd

default: Log

string: which output will be directed to stdout (standard out)

Log
: log messages

SAM
: alignments in SAM format (which normally are output to Aligned.out.sam file), normal standard output will go into Log.std.out

BAM_Unsorted
: alignments in BAM format, unsorted. Requires –outSAMtype BAM Unsorted

BAM_SortedByCoordinate
: alignments in BAM format, unsorted. Requires –outSAMtype BAM SortedByCoordinate

BAM_Quant
: alignments to transcriptome in BAM format, unsorted. Requires –quantMode TranscriptomeSAM

Filtering of the alignments

- STAR performs extensive filtering of the alignments by alignment score, mapped length, number of mismatches, and multimapping status.

- Only alignments that pass these filters are output into the SAM/BAM files.

- All the filtering conditions are combined with AND operations, i.e., all the conditions have to be satisfied for an acceptable alignment.

Alignment scoring

- For each of the putative alignments, STAR calculates the local alignment score, equal to the sum of +1/−1 for matched/mismatched nucleotides, as well as user-definable scores for insertions/deletions, genomic alignment length, and annotated splice junctions.

- For paired-end reads, alignment score is a sum of the scores for both mates. Alignment score can be saved as SAM attribute AS.

Alignment scoring

`--scoreGap` (=0 by default) splice junction penalty (independent on intron motif)

`--scoreGapNoncan` (=-8 by default) noncanonical junction penalty

`--scoreGapGCAG` (=-4 by default) GC/AG (CT/GC) junction penalty

`--scoreGapATAC` (=-8 by default) AT/AC (GT/AT) junction penalty

`--scoreGenomicLengthLog2scale` (=-0.25 by default) penalty logarithmically scaled with genomic length of the alignment: scoreGenomicLengthLog2scale*log2(genomic Length)

`--scoreDelOpen` (=-2 by default) deletion "open" penalty

`--scoreDelBase` (=-2 by default) deletion "extension" penalty per base (in addition to `--scoreDelOpen`)

`--scoreInsOpen` (=-2 by default) insertion "open" penalty

`--scoreInsBase` (=-2 by default) insertion "extension" penalty per base (in addition to `--scoreInsOpen`)

`--sjdbScore` 2 bonus score for alignments that cross-annotated junctions

Minimum alignment score

- To filter out poor alignments, users can define the minimum alignment score and the minimum number of matched bases:

`--outFilterScoreMinOverLread` (=0.66 by default): minimum alignment score normalized to read length

`--outFilterMatchNminOverLread` (=0.66 by default): minimum number of matched bases normalized to read length

- By default, valid alignments have to have score >0.66*readLength and number of matched bases >0.66*readLength. As always, the readLength is the sum of the mate's length for paired-end reads.

Paired-end alignments

- The mates of a paired-end read are the end sequences of one cDNA molecule (“insert”), and therefore STAR normally does not consider the mates separately, but rather treats the mates as end portions of one read.

- Following this logic, by default STAR only allows correctly (“concordantly”) paired alignments. Both single-end and non-concordantly paired alignments are considered invalid and are not output into the main alignment files.

Mismatches

`--outFilterMismatchNmax (=10 by default)`

Maximum allowed number of mismatches per alignment.

`--outFilterMismatchNoverLmax (=0.3 by default)`

Maximum allowed number of mismatches per alignment normalized to the **mapped** length is less than this value.

`--outFilterMismatchNoverReadLmax (=1 by default)`

Maximum allowed number of mismatches per alignment normalized to the **full** read length.

Mismatches

- For example, setting

`--outFilterMismatchNoverReadLmax 0.04`

will allow no more than 8 mismatches for 2 × 100 paired-end reads.

- Unless the “end-to-end” alignment is requested, STAR will trim (“soft-clip”) reads whenever the number of mismatches exceeds the above thresholds and may still be able to map the reads if the alignments satisfy minimum score and mapped length criteria.

- Note that mismatches are not counted in the trimmed (“soft clipped”) portion of the reads.

Soft-clipping

- STAR utilizes a “local alignment”-like strategy and tries to find the alignment with the best alignment score, rather than trying to map reads end-to-end (which is a common strategy in many popular RNA and DNA aligners).

- STAR will trim reads at the 5′ and 3′ ends in case such trimming gives a better alignment score than the end-to-end alignment with extra mismatches.

Soft-clipping

- There are several reasons for the trimming the ends, such as (1) poor sequencing quality of the tails, (2) adapter/polyA tails sequences, (3) and short splice overhangs.

- The trimming ("soft-clipping") of the read ends improves mapping accuracy (both sensitivity and precision) because:
  - Sequencing error rate increase toward the ends, and soft-clipping helps to map reads with poor quality tails—this is especially true for longer reads.
  - Soft-clipping allows to trim unwanted sequences at the end of the reads (adapters, A-tails, etc).

Soft-clipping

- Note that short splice overhangs (<10nt) are very hard to place correctly without a database of known junctions, and in many cases, STAR will soft-clip these short overhangs rather than mapping them to low confidence loci. Without soft-clipping allowed, a false end-to-end alignment with multiple mismatches will often win over, which may, for instance, yield erroneous expression of pseudogenes.

- In some situations, such as mapping short RNA data, the end-to-end alignments might be preferred, which can be done with `--alignEndsType EndToEnd` option.

Multimappers

- Reads that can be mapped to more than one genomic location with equally (or nearly equally) well are called “multimappers.”

- For each read STAR finds many putative alignments and calculates alignment scores for each of them. If the maximum score for a given read is maxScore, then all the alignments with scores >= maxScore-scoreRange are considered multimapping alignments for this read.

Multimappers

- The value of scoreRange is defined by input parameter `--outFilterMultimapScoreRange`. By default, this parameter is set to 1, which means that any alignment which has an extra mismatch compared to the best alignment will not be in the multimapping score range, since a mismatch reduces alignment score by 2.

Multimappers

- If the number of multimapping alignments for a read is less than `--outFilterMultimapNmax` (=10 by default), all of these alignments will be output into the main SAM/BAM files; otherwise, the read will be considered unmapped and none of the alignments will be output.

- `--outFilterMultimapNmax`: maximum number of loci the read is allowed to map to. Alignments (all of them) will be output only if the read maps to no more loci than this value.

- The number of alignments is reported in the “NH:i” SAM attribute.

Multimappers

- By default, only one of the multimapping alignments is considered primary; all others are marked as “secondary” (0 × 100 in the FLAG), even if they have the same score as the best alignment. This behavior can be changed by specifying `--outSAMprimaryFlag AllBestScore`, in which case all alignments with the score equal to the best are reported as primary, while all lower score multimapping alignments are marked with 0 × 100.

Tuning mapping sensitivity

- `--seedSearchStartLmax` (=50 by default): This parameter defines the maximum length of the blocks the read is split into by seed search start points. Reducing this parameter will increase the overall sensitivity of mapping, including annotated and unannotated junctions, indels, multiple mismatches, and other complicated cases. The effect will be especially pronounced in cases of poor sequencing quality or mapping to a divergent genome. It is recommended that this parameter is reduced for reads shorter than 50nt and is set at ½ to ¾ of the read length.

Tuning mapping sensitivity

- `--seedSearchStartLmaxOverLread` (=1.0 by default): This parameter has the same meaning but is normalized to the read length. The shorter of the two parameters will be utilized.

- `--winAnchorMultimapNmax` (=50 by default): This parameter defines the maximum number of loci anchor seeds can be mapped to. Increasing this parameter allows for shorter anchor seeds, which increases the search space and improves the mapping accuracy. However, this improvement in accuracy comes at the cost of reduced mapping speed.

2-Pass mapping

- To increase mapping accuracy (especially the sensitivity to unannotated splices), STAR can be run in the 2-pass mode. The 1st pass serves to detect novel junctions, and in the 2nd pass, the detected junctions are added to the annotated junctions, and all reads are re-mapped to finalize the alignments.

- While this procedure does not significantly increase the number of novel collapsed junctions, it substantially increase the number of reads crossing the novel junctions, by allowing novel splices with shorter overhang.

- This procedure is especially advantageous in cases where annotations are unavailable or incomplete.

Multi-sample 2-pass mapping

- For a study with multiple samples, it is recommended to collect 1st pass junctions from all samples.

1. Run 1st mapping pass for all samples with "usual" parameters. Using annotations is recommended either at the genome generation step, or mapping step.

2. Run 2nd mapping pass for all samples, listing SJ.out.tab files from all samples in `--sjdbFileChrStartEnd /path/to/sj1.tab /path/to/sj2.tab ....`

Per-sample 2-pass mapping

- Annotated junctions will be included in both the 1st and 2nd passes. To run STAR 2-pass mapping for each sample separately, use

```
--twopassMode Basic
```

- STAR will perform the 1st pass mapping, then it will automatically extract junctions, insert them into the genome index, and, finally, re-map all reads in the 2nd mapping pass.

- This option can be used with annotations, which can be included either at the run-time, or at the genome generation step.

Post-mapping processing

- The standard output of STAR mapping consists of alignments in SAM/BAM files and the list of detected splice junctions `SJ.out.tab`.

Remove duplicates

- STAR can remove duplicate reads starting from coordinate-sorted BAM file with the following command:

```
STAR --runMode inputAlignmentsFromBAM --
inputBAMfile Aligned.sortedByCoord.out.bam --
bamRemoveDuplicatesType UniqueIdentical
```

- The reads are considered duplicates if their alignment starts (after extending soft-clipped bases) and CIGARS (i.e., indels and junctions) coincide.

Transcriptomic output

- With `--quantMode TranscriptomeSAM` option, STAR will output alignments translated into transcript coordinates in the `Aligned.toTranscriptome.out.bam` file (in addition to alignments in genomic coordinates in `Aligned.*.sam/bam` files).

- These transcriptomic alignments can be used by various transcript quantification software that require reads to be mapped to transcriptome, such as RSEM or eXpress.

Counting number of reads

- With `--quantMode GeneCounts` option, STAR will count number of reads per gene while mapping.

- A read is counted if it overlaps (by 1 or more nucleotides) one and only one gene.

- Both ends of the paired-end read are checked for overlaps. The counts coincide with those produced by htseq-count with default parameters. This option requires annotations (GTF or GFF with `--sjdbGTFfile` option) at the genome generation step or at the mapping step.

Counting number of reads

- STAR outputs read counts per gene into `ReadsPerGene.out.tab` file with four columns, with the columns 2–4 corresponding to different strand options:
  1. Gene ID.
  2. Counts for unstranded RNA-seq.
  3. Counts for the stranded RNA-seq with the 1st read strand matching the RNAstrand (htseq-count option -s yes).
  4. Counts for the stranded RNA-seq with the 2nd read strand matching the RNAstrand (htseq-count option -s reverse).

Counting number of reads

- With `--quantMode GeneCounts TranscriptomeSAM`, STAR will generate both the `Aligned.toTranscriptome.out.bam` and `ReadsPerGene.out.tab` outputs.

Example: PE

```
sample_PE='SRR3191545'

STAR --runThreadN 5 --runMode alignReads \

--genomeDir /home/username/reference/human/star --readFilesIn
/home/Data/zika/${sample_PE}_1.fastq.gz
/home/Data/zika/${sample_PE}_2.fastq.gz --readFilesCommand zcat \

--outFileNamePrefix /home/username/star/${sample_PE}_ --
outFilterMultimapNmax 1 –outSAMtype BAM Unsorted \

--outSAMattributes All --quantMode GeneCounts --sjdbGTFfile
/home/reference/human/Homo_sapiens.GRCh38.108.gtf \

--twopassMode Basic
```

Example: PE

```
(base) cat run_star_pe.example.sh
#!/bin/bash
set -e
set -u
set -o pipefail

sample_PE='SRR3191545'

STAR --runThreadN 10 --runMode alignReads \
        --genomeDir /home/yjjeong/reference/human/star \
        --readFilesIn /home/Data/zika/${sample_PE}_1.fastq.gz /home/Data/zika/${sample_PE}_2.fastq.gz \
        --readFilesCommand zcat \
        --outFileNamePrefix /home/yjjeong/star/${sample_PE}_ \
        --outFilterMultimapNmax 1 --outSAMtype BAM Unsorted \
        --outSAMattributes All --quantMode GeneCounts \
        --sjdbGTFfile /home/reference/human/Homo_sapiens.GRCh38.108.gtf \
        --twopassMode Basic
```

Example: SE

```
sample_SE='SRR3194431'

STAR --runThreadN 5 --runMode alignReads \

--genomeDir /home/username/reference/human/star \

--readFilesIn /home/Data/zika/${sample_SE}_1.fastq.gz --readFilesCommand zcat \

--outFileNamePrefix /home/username/star/${sample_SE}_ --outFilterMultimapNmax 1
--outSAMtype BAM Unsorted \

--outSAMattributes All --quantMode GeneCounts --sjdbGTFfile /
home/reference/human/Homo_sapiens.GRCh38.108.gtf \

--twopassMode Basic
```

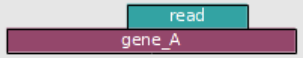
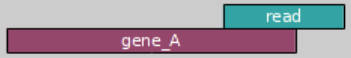
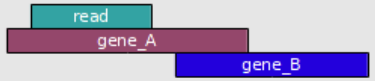
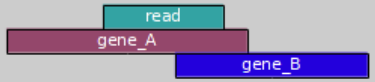
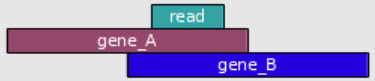
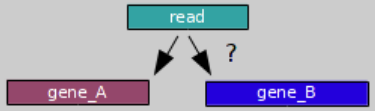
Example: SE

```
(base) cat run_star_se.example.sh
#!/bin/bash
set -e
set -u
set -o pipefail

sample_SE='SRR3194431'

STAR --runThreadN 10 --runMode alignReads \
        --genomeDir /home/yjjeong/reference/human/star \
        --readFilesIn /home/Data/zika/${sample_SE}_1.fastq.gz --readFilesCommand zcat \
        --outFileNamePrefix /home/yjjeong/star/${sample_SE}_ --outFilterMultimapNmax 1 --outSAMtype BAM Unsorted \
        --outSAMattributes All --quantMode GeneCounts --sjdbGTFfile /home/reference/human/Homo_sapiens.GRCh38.108.gtf \
        --twopassMode Basic
```

htseq-count

	union	intersection_strict	intersection_nonempty
	gene_A	gene_A	gene_A
	gene_A	no_feature	gene_A
	gene_A	no_feature	gene_A
	gene_A	gene_A	gene_A
	gene_A	gene_A	gene_A
	ambiguous (both genes with --nonunique all)	gene_A	gene_A
	ambiguous (both genes with --nonunique all)		
	alignment_not_unique (both genes with --nonunique all)		

http://htseq.readthedocs.io/en/master/count.html

htseq-count

```
sample_list=('SRR3191542' 'SRR3191543' 'SRR3191544' 'SRR3191545' 'S
RR3194428' 'SRR3194429' 'SRR3194430' 'SRR3194431')

num_sample=${#sample_list[@]}


for (( i=0;i<$num_sample;i++ )); do

      htseq-count -m union -s no -a 0 -i gene_id -f bam /home/
username/star/${sample_list[${i}]}_Aligned.out.bam \

/home/reference/human/Homo_sapiens.GRCh38.108.gtf >
/home/username/htseq/${sample_list[${i}]}.htseqcount.txt

done
```

htseq-count

```
conda activate python-3.6
```