Chapter 13

Optimizing RNA-Seq Mapping with STAR

Alexander Dobin and Thomas R. Gingeras

Abstract

Recent advances in high-throughput sequencing technology made it possible to probe the cell transcriptomes by generating hundreds of millions of short reads which represent the fragments of the transcribed RNA molecules. The first and the most crucial task in the RNA-seq data analysis is mapping of the reads to the reference genome. STAR (Spliced Transcripts Alignment to a Reference) is an RNA-seq mapper that performs highly accurate spliced sequence alignment at an ultrafast speed. STAR alignment algorithm can be controlled by many user-defined parameters. Here, we describe the most important STAR options and parameters, as well as best practices for achieving the maximum mapping accuracy and speed.

Key words Sequence alignment, Reads mapping, RNA-seq, Transcriptome, Spliced alignment, STAR

1 Introduction

Sequencing of transcribed RNA molecules (RNA-seq) is an invaluable tool for studying cell transcriptomes at high resolution and depth. RNA-seq datasets typically consist of tens to hundreds of millions of relatively short (30–200 nt) sequence fragments of the original RNA transcripts. The very first step in a typical RNA-seq analysis pipeline is mapping (alignment) of the short reads to a reference genome. The large number of reads, as well as large genome sizes of many important species, makes this task very computationally intensive. The RNA transcripts are often spliced, requiring mapping to noncontiguous regions of the genome. This creates a unique challenge for the RNA-seq mapping, both in terms of speed and accuracy. The STAR [1] RNA-seq mapper was developed to overcome these challenges and enable highly accurate spliced reads alignment at ultrafast speed. STAR is feature-rich software, capable of detecting annotated and novel splice junctions, as well as chimeric and circular RNA. Because of its ability to map spliced sequences of any length with moderate error rate, STAR provides scalability for emerging sequencing technologies. In addition to standard SAM/BAM output, STAR can generate various other data files useful for downstream analyses such as transcript/gene expression quantification, differential gene expression, novel isoform reconstruction, signal visualization, etc.

Here, we describe many important parameters and options that can be tweaked to optimize STAR performance, both for mapping accuracy and speed. Subsection 2 describes the required software, hardware, and input files. Subsection 3.1 describes a generation of genomic indexes, including the basic command and advanced options. Subsection 3.2 describes the mapping of the reads to the reference genome, including the input reads files options, controlling the output, tuning mapping sensitivity, filtering of alignments and splice junctions, and 2-pass mapping procedure. Subsection 3.3 describes various post-mapping processing, including generating wiggle files, removing duplicates, and converting to transcriptomic coordinates.

2 Materials

2.1 Hardware

STAR requires a computer with 64-bit Unix, Linux, or Mac OS X operating system.

Maximum required RAM (random access memory) depends on the genome size, approximately 10×GenomeSize bytes. For instance, for human genome a minimum of 32GB of RAM is recommended.

Both input FASTQ files and output SAM/BAM files require large disk space (>100GB recommended).

STAR is multithreaded, i.e., it can be run on multiple execution threads. The number of threads is defined by `--runThreadN <number-of-threads>` option. Typically, this number should be equal to the number of processor cores.

2.2 Software

Latest release of STAR software can be downloaded from https://github.com/alexdobin/STAR/releases. The releases include the pre-compiled STAR executables for Linux and Mac OS X, as well as instructions on how to compile STAR from the source code.

Question about STAR usage should be asked on the STAR user discussion group: https://groups.google.com/forum/#!forum/rna-star.

STAR manual contains the most up-to-date information:

https://github.com/alexdobin/STAR/raw/master/doc/STARmanual.pdf

2.3 Input Files

For generating genome indexes, the FASTA file(s) with reference genome sequence(s) is needed, as well as a file containing annotations (annotated transcripts) in GTF or GFF3 formats.

The input RNA-seq reads can be in the FASTQ format (standard for many current high-throughput sequencers) or in the FASTA format.

3 Methods

STAR can be run with multiple parameters (options), which have the following general form:

```
--parameterName  parameterValue1  [parameter-
Value2] <…>
```

Parameter names always start with double dashes and are followed by one or multiple parameter values separated by spaces.

Typical STAR workflow consists of two major steps: (1) generating genome indexes and (2) mapping the RNA-seq reads, which are described in Subsection 3.1 and 3.2, respectively.

3.1 Generating Genome Indices

3.1.1 Basic Command

Basic command to generate genome indexes is as follows:

```
STAR --runMode genomeGenerate --genomeDir /path/
to/genome/directory/ --genomeFastaFiles /path/to/
seq1.fasta /path/to/seq2.fasta --runThreadN 12
```

```
--genomeDir /path/to/genome/directory/
```

This option specifies the path to the genome directory, which will contain all the genome files. Each genome should be generated in a separate directory, and the directory name serves as a unique identifier of the genome. The directory should be created before the STAR run.

```
--genomeFastaFiles /path/to/seq1.fasta /path/to/
seq2.fasta
```

One or multiple files containing genome (reference) sequences in the FASTA format. Each file may contain one or more sequences. Multiline FASTA format is supported. Lower or upper case characters are treated the same. Aa, Cc, Gg, and Tt are considered nucleotides, while all other characters are converted to N (i.e., nucleotide unknown).

```
--runThreadN 12
```

Number of threads to be used.

3.1.2 Including Annotations

STAR uses annotations to extract known splice junctions and then builds “spliced” sequences by deleting intron sequences, i.e., joining the sequences of the exons. This is highly recommended, since it allows for a more accurate mapping of the spliced reads, especially those with very short junction overhangs (<10 nt). The first option to supply annotations is in the form of the GTF or GFF file:

```
--sjdbGTFfile /path/to/annotations.gtf
```

Another option is to supply the file which defines splice junctions loci

```
--sjdbFileChrStartEnd /path/to/junctionLoci.tab
```

which has four tab-separated columns: *Chromosome <tab> Start <tab> End <tab> Strand*. Here *Start* and *End* are the started and end coordinates of the junction introns.

Importantly, chromosome names in the GTF and the junction Loci files should coincide with the chromosome names in the genome sequence FASTA files.

Annotations can also be included on the fly at the mapping stage (*see* Subsection 3.2.2).

Selecting --sjdbOverhang

`--sjdbOverhang (=100 by default)` is an important parameter which defines the number of bases taken from both (donor and acceptor) sides of the junction that are joined together to form an additional "junction" sequences. At the mapping stage, the reads are aligned to both genomic and these junction sequences simultaneously. If a read maps to one of the junction sequences and crosses the junction in the middle of it, the coordinates of two spliced pieces are translated back to genomic space and added to the collection of seeds, which are then all "stitched" together to form the final alignment.

In general, the default value of 100 is acceptable for reads longer than 50 nucleotides. Strictly speaking, the best sensitivity for detection of annotated junctions is achieved by setting `--sjdbOverhang` *`readLength-1`*, where *`readLength`* is the read length (one end/mate length for paired-end reads). This value is ideal to map a read that has *readLength-1* nucleotides on one side of the junction and 1 nucleotide on the other. However, in the process of "maximal mapped length" search, the read is split into pieces of no longer than `--seedSearchStartLmax (=50 by default)` nucleotides (*see* Subsection 3.2.6); hence, even if the read (mate) is longer than `--sjdbOverhang`, it can still be mapped to the junction sequence, as long as `--sjdbOverhang > --seedSearchStartLmax`. At the same time, if `--sjdbOverhang` is too long, more seeds will be multimapping, since the junction sequences are redundant with the genome sequence. Then again, STAR transforms the seed coordinates from junction to genome coordinates, and equivalent seeds are collapsed; thus, in the end, it affects only a marginal population of reads.

3.1.3 Advanced Parameters

`--genomeSAindexNbases (=14 by default)` is the size of N-mers used for preindexing of the suffix array which speedups the search. The genomic N-mer locations in the suffix array are stored in the SAindex file in the genome directory. By default, N=14, which means 4^14= 268,435,456 N-mers are stored. Small genomes do not have that many different N-mers; hence, this parameter needs to be scaled down to ~min(14, log2(GenomeLength)/2 − 1). This is an approximate formula since it depends on the actual N-mer sequence content. The mapping results do not depend on `--genomeSAindexNbases`, but larger values increase mapping speed.

`--genomeChrBinNbits (=18 by default)` defines the "padding" size (log2) of the reference sequences. For a genome with a large (>5,000) number of references (chromosomes/scaffolds), `--genomeChrBinNbits` needs to be reduced to decrease RAM consumption. The following scaling is recommended:

`--genomeChrBinNbits = min(18, log2(GenomeLength/NumberOfReferences))`. For example, for 3 gigaBase genome with 100,000 chromosomes/scaffolds, this is equal to 15.

3.2 Mapping Reads to the Genome

3.2.1 Basic Command

Basic command to map read to the genome is as follows:

```
STAR --genomeDir /path/to/genome/directory/ --read
FilesIn read1.fastq [read2.fastq] --runThreadN 12
```

The path to the genome directory, where the genome indices where generated (*see* Subsection 3.1.1):

```
--genomeDir /path/to/genome/directory/
```

The input read sequence(s) in the FASTQ or FASTA format:

```
--readFilesIn read1.fastq [read2.fastq]
```

Number of threads (parallel processes) to be used:

```
--runThreadN 12
```

3.2.2 Including Annotations at the Mapping Step

Similar to including annotations at the genome generation step (Subsection 3.1.2), annotations can be included at the mapping step by specifying `--sjdbGTFfile /path/to/annotations.gtf` and/or `--sjdbFileChrStartEnd /path/to/junctionLoci.tab`.

The junctions in these files will be added on-the-fly to the junctions that were included at the genome generation step. If `--sjdbOverhang` parameters (Subsection 3.1.2.1) are 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.

3.2.3 Input Options

Input Files

STAR can read from multiline FASTA files and single-line FASTQ files.

For single-end reads, only one FASTQ or FASTA file has to be specified:

```
--readFilesIn read.fastq
```

For paired-end reads, two files separated by space have to be specified:

```
--readFilesIn read1.fastq read2.fastq
```

Multiple input files can be specified in comma-separated list, e.g.:
Single-end reads:

```
--readFilesIn A.fastq,B.fastq,C.fastq
```

Spaces are not allowed in this comma-separated list, unless they are in double quotes, e.g., “A A.fastq.”
Paired-end reads:

```
--readFilesIn A1.fastq,B1.fastq,C1.fastq A2.fastq,
  B2.fastq,C2.fastq
```

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

```
--readFilesCommand <pre-processing command>
```

specifies the OS command to preprocess the reads. This command should take the file name as input parameter and stream reads in text format into standard output. The command may be a built-in OS command with multiple options or a user script. For example, both of the following options can be used to unzip gzipped FASTQ files:

```
--readFilesCommand zcat
--readFilesCommand gunzip -c
```

Trimming the Read Sequences

Several parameters can be used to perform basic trimming of the input sequences. All of these commands accept one or two values. For paired-end reads, the two values are used for read1 and read2, but if only one value is given, it will be assumed the same for both reads:

`--clip3pNbases (=0 by default)`

Number(s) of bases to trim from the 3′ end of the read(s).

`--clip5pNbases (=0 by default)`

Number(s) of bases to trim from the 5′ end of the read(s).

`--clip3pAdapterSeq (=- i.e. none by default)`

Adapter sequence(s) to be trimmed from the 3′ end of the read(s).

`--clip3pAdapterMMp (=0.1 by default)`

Maximum proportion(s) of mismatches for 3′ adapter trimming.

`--clip3pAfterAdapterNbases (=0.1 by default)`

Number(s) of bases to clip from 3′ end(s) after trimming the adapter sequence.

3.2.4 Controlling Output of Alignments

Sorted and Unsorted SAM and BAM

By default, the alignments are output in the text SAM format into the `Aligned.out.sam` file. The mate alignments for paired-end reads are adjacent to each other; all multimapping alignments are output consecutively. STAR can also output alignments directly in the BAM format, as well as BAM sorted by coordinate using the following option:

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

The 1st word of this option is `SAM`, `BAM`, or `None`; the 2nd word can be `Unsorted` or `SortedByCoordinate`. Unsorted and SortedByCoordinate options can also be used simultaneously.

For coordinate-sorted BAM output, STAR allocates RAM (dynamic memory) for sorting after the mapping is completed. The amount of allocated RAM can be specified by `--limitBAMsortRAM <bytes>` parameter. By default, this parameter is set to 0, which allocates the same amount of memory for sorting as was used for mapping (i.e., the size of the genome indices). If shared memory is used with `--genomeLoad` options (*see* Subsection 3.2.9), the `--limitBAMsortRAM` has to be specified explicitly.

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.

Attributes

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

```
--outSAMattributes <list-of-attributes>
```

The list contains the two-character SAM attributes including:

`NH`	number of loci a read maps to (=1 for unique mappers, >1 for multimappers)
`HI`	index for the multimapping alignments (starts with 1, =1 for unique mappers)
`AS`	alignment score (*see* Subsection 3.2.5)
`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, see [2] and SAM specifications

`jM jM:B:c,M1,M2,…`

List of intron motifs for all junctions (i.e., N operations in CIGAR) 0-noncanonical; 1-GT/AG; 2-CT/AC; 3-GC/AG; 4-CT/GC; 5-AT/AC; 6-GT/AT. Note that intron motif here is determine always with respect to the (+) strand. To indicate annotated splice junctions, 20 is added to the intron motif numbers:

`jI    jI:B:I,Start1,End1,Start2,End2,…`

Starts/ends of introns for all junctions.

`XS` strand attribute for spliced alignments.

The `nM` tag is different from the standard `NM`: `nM` is the number of mismatches per pair (not per mate), and it does not include the indels (i.e., it is not edit distance per mate like `NM`).

`jM jI` attributes require samtools 0.1.18 or later and may be incompatible with some downstream tools.

`--outSAMattributes` can also accept the following options: `Standard` for (`NH HI AS nM`), `All` (for `NH HI AS nM NM MD jM jI`), and `None`.

SAM Read Groups

Read groups can be added to SAM/BAM records while mapping using `--outSAMattrRGline <RG line>`. The read group fields should be separated by space, and the first field should be "ID:<rg-id>." If field values contain spaces, they should be double quoted, e.g., `--outSAMattrRGline ID:zzz "DS:z z."` The entire string will be added as `@RG` line in the SAM header, and the `ID` field will be added as attribute to each alignment.

If multiple files were supplied as comma-separated list in `--readFilesIn` (*see* Subsection 3.2.3), corresponding read group entries may be supplied as a comma-separated list as well, e.g., `--outSAMattrRGline ID:sampleA CN:AA DS:AAA, ID:sampleBB CN:bb DS:bbbb, ID:sampleC CN:ccc DS:cccc`. Note that in this list commas have to be surrounded by spaces. This list will be split into multiple `@RG` lines in the SAM header, and the reads from different input files will be given matching read group `IDs`.

Output File Name Prefix

By default, STAR will write all output files in the current working directory. This can be changed with the `--outFileNamePrefix /path/to/output/prefix/` option which will add the specified prefix to all output file names.

Standard Output

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

```
--outStd    Log
```

This option controls which output will be directed to stdout (standard-out) stream.

`Log` logging messages
`SAM` alignments in SAM format (which normally are output to Aligned.out.sam file)
`BAM_Unsorted` unsorted BAM with `--outSAMtype BAM Unsorted`
`BAM_SortedByCoordinate` coordinate-sorted BAM with `--outSAMtype BAM SortedByCoordinate`
`BAM_Quant` unsorted transcriptome alignments with `--quantMode TranscriptomeSAM`

Temporary Output Directory

STAR writes temporary files into a temporary output directory, which, by default, is `_STARtmp` within the STAR mapping directory. If the `--outFileNamePrefix` option is used, the temporary directory is `outFileNamePrefix_STARtmp`. The `--outTmpDir </path/to/tmp/dir/>` option can be used to change the location of the temporary directory, which might increase the speed in cases where large temporary files are written into this directory, e.g., sorted BAM output. For instance, if the mapping directory (where the final output files will be stored) is on a slow network drive, the `--outTmpDir` may be pointed to a much faster local drive.

3.2.5 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 (see below) 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 (*see* Subsection 3.2.4).

`--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

(*See* Subsection 3.1.2).

Minimum Alignment Score and Length

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

`--outFilterScoreMin` (=0 by default) minimum alignment score

`--outFilterMatchNmin` (=0 by default) minimum number of matched bases

The same filtering condition can also be specified with normalization over the read length (sum of the mates' lengths for paired-end reads):

`--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

Note that the four conditions are combined with the AND operation, i.e., the most stringent condition determines whether an alignment is valid. 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.

In principle, unpaired alignments can be output into the main SAM/BAM files by reducing `--outFilterMatchNminOverLread` and `--outFilterScoreMinOverLread` to below 0.5. However, the unpaired alignments typically contain a large number of false positives, and their usage is strongly discouraged except for detecting chimeric (fusion) transcripts. The non-concordant pairs can be output in the separate *Chimeric.out.sam* file if chimeric detection is switched on.

Mismatches

The maximum number of mismatches is controlled by tow parameters (combined with the AND operation, as always):

`--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.

The default value of `--outFilterMismatchNmax` 10 mismatches in STAR is quite arbitrary and needs to be adjusted in each particular situation. All of these parameters relate to the total number of mismatches in the paired alignment (i.e., sum of two mates). For example, setting `--outFilterMismatchNoverReadLmax 0.04` will allow no more than 8 mismatches for 2×100 paired-end reads.

The mismatches can be caused by sequencing errors, SNPs, and RNA editing, which may require setting high thresholds in some cases. However, unless the "end-to-end" alignment (*see* Subsection 3.2.5) 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.

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:

1. 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.
2. Soft-clipping allows to trim unwanted sequences at the end of the reads (adapters, A-tails, etc).
3. Note that short splice overhangs (~<10 nt) are very hard to place correctly without a database of known junctions (*see* Subsection 3.1.2), 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." STAR defines and outputs multimappers using the following rules.

For each read STAR finds many putative alignments and calculates alignment scores for each of them (*see* Subsection 3.2.5). 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. 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. 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.

The number of alignments is reported in the "NH:i" SAM attribute (*see* Subsection 3.2.4). 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.

3.2.6 Tuning Mapping Sensitivity

The main parameters that can be tweaked to improve accuracy of the alignments are:

`--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 50 nt and is set at ½ to ¾ of the read length:

`--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. Decreasing 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.

In general, the following strategy for tuning mapping parameter to achieve higher accuracy is recommended:

1. Choose a good metric for false positives and false negatives; for instance, annotated splice sites can be considered pseudo-true positive, while unannotated ones pseudo-false positive.
2. Map one or a few representative samples under study tuning several selected parameters, and calculate the sensitivity and precision using the selected metrics. This will yield a pseudo-ROC curve that can be used to select the best parameters based on your preference for sensitivity and precision.

3.2.7 Filtering Splice Junctions

Detection of spliced alignments is the crucial task in mapping RNA-seq data. STAR has a number of parameters that control the filtering of the spliced reads and junctions which can be used to optimize the accuracy of the splice junction detection and allows for tradeoff between sensitivity and precision.

Filtering Introns

The following filters control the introns of the alignments for each read:

`--alignIntronMin (=21 by default)`

defines the minimum intron size, i.e., the genomic gap in the alignments is considered splice junction intro if the gap length is bigger than `--alignIntronMin`; otherwise, the gap is considered "deletion":

`--alignIntronMax (=0 by default)`

defines maximum intron size: alignments with larger gaps are considered chimeric and are not allowed in the normal output. The default 0 sets this parameter to `(2^winBinNbits)*winAnchorDistNbins=589,824`

`--alignMatesGapMax (=0 by default)`

Similarly to the `--alignIntronMax`, this parameter defines the maximum gap between two mates. The gap between the mates may contain one or more splice junctions, and thus it is expected to be larger or equal than the `--alignIntronMax`.

`--alignSJoverhangMin (=5 by default)`

Sequence of a spliced read is split on two sides of the splice junction. This parameter defines the minimum allowed length of this sequence (overhang). The alignments with short splice overhangs are less reliable since short sequences may map to many loci.

`--alignSJDBoverhangMin (=3 by default)`

This parameter is similar to the `--alignSJoverhangMin`; this parameter defines the minimum overhang but only for the annotated junctions (*see* Subsection 3.1.2). While annotated junctions are considered more reliable than unannotated ones, the very short splice overhangs may nevertheless yield false splices.

`--outFilterIntronMotifs (=None by default)`

This parameter controls the intron motifs of the spliced alignments. The GT/AG, GC/AG, and CT/AC are considered canonical motifs, while all others are noncanonical.

`RemoveNoncanonical` prohibits alignments that contain **any** splice junctions with noncanonical intron motifs.

`RemoveNoncanonicalUnannotated` prohibits alignments that contain **unannotated** splice junctions with noncanonical intron motifs.

Filtering Output to SJ.out.tab

STAR collects information from spliced reads that supports a particular junction and outputs it into the SJ.out.tab file. Each line in this file contains information about one splice junction, which is crossed by one or many spliced reads. While there are many millions of spliced reads, there are only a few hundred thousands of "collapsed" junctions. The parameters described below control filtering of splice junctions into the SJ.out.tab file, creating a highly trustworthy set of junctions. Unlike the filters in Subsection 3.2.7, these filters work not on individual splices reads but rather on collapsed (aggregated) splice junctions. Only unannotated junctions are affected by these filters; all annotated junctions are output into SJ.out.tab without filtering. The filtering depends on the junction intron motifs; by default, the noncanonical junctions are considered less trustworthy and thus require much more stringent filters.

`--outSJfilterCountUniqueMin (=3 1 1 1 by default)`

The four numbers define the minimum numbers of uniquely mapped reads that support each junction for different junction intron motifs: (1) noncanonical, (2) GT/AG, (3) GC/AG, (4) AT/AC. By default, noncanonical junctions require at least three unique reads per junction, while all noncanonical motifs require only 1 unique read per junction. Increasing these numbers increases the precision (i.e., decreases false discovery rate) for the junction detection while at the same time decreasing the sensitivity (i.e., increasing the false-negative rate).

`--outSJfilterCountTotalMin (=3 1 1 1 by default)`

Same as before, but both unique and multimapping reads are counted. Note that junctions pass these two filters if either of the --outSJfilterCountUniqueMin OR --outSJfilterCountTotalMin conditions is satisfied.

`--outSJfilterOverhangMin (=30 12 12 12 by default)`

The four numbers define the minimum splice overhangs for reads supporting each junction for different junction intron motifs. This means that at least one read should have an overhang of at least 30 nt for noncanonical junctions and 12 nt for all canonical junctions.

`--outSJfilterDistToOtherSJmin (=10 0 5 10 by default)`

The four numbers define the minimum distance from the junction donor/acceptor sites to other junction sites for different junction intron motifs. This means junction acceptor/donor sites should be at least 10 nt away from other junctions for the noncanonical junctions, 0 nt, for GT/AG motifs; 5 nt, for GC/AG; and 10 nt, for AT/AC. This parameter prevents output of non-reliable junctions with donor/acceptor sites only shifted from the other junctions,

`--outSJfilterIntronMaxVsReadN (=50000 100000 200000 by default)`

The numbers define maximum gap (intron) allowed for junctions supported by 1,2,3,... reads. By default, junctions supported by 1 read can have gaps <=50,000 nt, by 2 reads, <=100,000 nt, and by 3 reads, <=200,000 nt; junctions supported by >=4 reads can have any gap, limited only by `--alignIntronMax`. This parameter is used to prevent rare long spliced alignments, i.e., those with long gaps and very few reads supporting them.

By default, the `--outSJfilter*` parameters describe above do not affect alignments in the SAM/BAM output files, but only filter the junctions output into `SJ.out.tab`. However, with `--outFilterType BySJout` option, alignments in the SAM/BAM output will only be allowed to cross the junctions which pass the filtering into `SJ.out.tab`.

This option makes SAM/BAM output files consistent with `SJ.out.tab` file.

Below we explain the logic of the splice junction filtering using example

options `--alignSJDBoverhangMin` 3 `--alignSJoverhangMin` 5 `--outSJfilterOverhangMin` 30 12 12 12 `--outFilterType BySJout`.

First, we filter all the alignments with very short overhangs, `--alignSJDBoverhangMin` 3 `--alignSJoverhangMin` 5.

Next, we create a confident set of junctions by requiring that at least one supporting read has a large enough overhang >= --outSJfilterOverhangMin 30 12 12 12 (i.e., 12 for unannotated canonical motifs or 30 for noncanonical).

Finally, with --outFilterType BySJout we prohibit any alignments across the junctions that did not make into the confident set.

For example, consider an unannotated GT/AG junction that is crossed by three spliced reads, with overhangs 6, 9, and 15 nt.

This junction will be output into the `SJ.out.tab` file with read count of 3 and the maximum overhang of 15. Also, all three splices will be reported in the SAM/BAM output. On the other hand, if the three overhangs were 6, 9, and 11 nt, the junction would not make it into the `SJ.out.tab`, because the maximum overhang is 11 which is less than the required 12. All three three alignments will be reported in the SAM/BAM output by default; however, if `--outFilterType BySJout` is used, those three splices would not be allowed in the SAM/BAM output.

3.2.8 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 multi-sample study, the best practice is to collect the junctions from all the samples after the 1st pass and use the full set of junctions for mapping each of the samples in the 2nd pass:

1. Run 1st mapping pass for all samples with normal parameters. Using annotations is highly 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/sample1/SJ.out.tab /path/to/sample2/SJ.out.tab ….`

Before starting the 2nd pass mapping, STAR will on-the-fly insert the junctions from the 1st pass into the genome indices. This approach yields the best and uniform sensitivity for novel junctions across all samples.

Per-sample 2-Pass Mapping

For studies containing single or incompatible samples, it is possible to run the 2-pass mapping with a single STAR command `--twopassMode Basic`. STAR will perform the 1st pass mapping, and then it will automatically extract junctions, insert them into the genome index, and, finally, re-map all reads in the 2nd mapping pass. Using the per-sample 2-pass approach yields slightly poorer sensitivity than the multi-sample 2-pass (Subsection 3.2.8). For instance, if a novel junction is highly expressed in only one sample and weakly (only a few reads with short overhang) in other samples, the per-sample 2-pass approach may only detect this junction in the former sample. On the other hand, the multi-sample 2-pass strategy will detect this junction in all samples.

3.2.9 Loading Genome into Shared Memory

The `--genomeLoad` parameter controls how the genome is loaded into memory. With `--genomeLoad LoadAndKeep`, STAR loads the genome as a standard Linux shared memory piece. Before loading the genome, STAR will check if the genome has already been loaded into the shared memory. The genomes are identified by their unique directory paths. If the genome has not been loaded, STAR job will load it and will keep it in memory even after STAR job itself finishes. The genome will be shared with all the other STAR instances. The genome can be removed from the shared memory running STAR with `--genomeLoad Remove`. The shared memory piece will be physically removed only after all STAR jobs attached to it are complete. With `--genomeLoad LoadAndRemove`, STAR will load genome in the shared memory and mark it for removal, so that the genome will be removed from the shared memory once all STAR jobs using it exit. If `--genomeLoad LoadAndExit`, STAR will load genome in the shared memory and immediately exit without performing any alignment, keeping the genome loaded in the shared memory for the future runs.

To check or remove shared memory pieces manually, the standard Linux command *`ipcs`* and *`ipcrm`* can be used. If the genome residing in shared memory is not used for a long time, it may get paged out of RAM which will slow down STAR runs considerably. It is strongly recommended to regularly reload (i.e., remove and load again) the shared memory genomes.

If `--genomeLoad NoSharedMemory`, shared memory is not used. This option is recommended if the shared memory is not configured properly on your server.

3.3 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`. STAR is also capable of generating other types of files as described below.

3.3.1 Wiggle Files

Wiggle files are useful for visualization of the RNA-seq signal on the genomic browsers such as UCSC genomic browser (http://genome.ucsc.edu/) or IGV browser (https://www.broadinstitute.

org/igv/). The signal represent the number reads crossing each genomic base. These options require `--outSAMtype BAM SortedByCoordinate`. STAR will generate separate signal files for uniquely mapping reads and unique + multimapping reads. In the latter case, the contribution of multimappers will be divided by the number of loci they map to.

`--outWigType (= None by default)`

Defines the type of signal output.

The 1st word can be `bedGraph` (for "bed-graph" formatting, see http://genome.ucsc.edu/goldenpath/help/bedgraph.html) or `wiggle` (for "wiggle" formatting, see http://genome.ucsc.edu/goldenpath/help/wiggle.html).

If the 2nd word is not present, the signal is generated from all the read bases. If 2nd word is `read1_5p`, the signal is generated only form 5′ of the 1st read, which is useful for CAGE/RAMPAGE data.

If the 2nd word is `read2`, the signal is generated only from the 2nd mate of the paired-end reads.

`--outWigStrand (=Stranded by default)`

Whether to output `stranded` or `untranded` signal

`--outWigNorm (=RPM by default)`

Type of the signal normalization:

`None` : no normalization, "raw" counts

`RPM` : normalize by the millions of mapped reads (i.e., divide by the total number of reads and multiply by 10^6). For the unique signal, the total number of reads includes only unique reads, while for unique + multiple, it includes both unique and multiple reads.

In addition to generating wiggle files at the mapping step, it can also be done using the previously mapped reads stored in coordinate-sorted BAM file, using `--inputBAMfile </path/to/aligned.bam> option, e.g.:`

```
STAR --inputBAMfile Aligned.sortedByCoord.out.bam --outWigType wiggle --outWigStrand Unstranded --outWigNorm None
```

3.3.2 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.

3.3.3 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 [3] or eXpress [4]. Note that STAR first aligns reads to entire genome and only then searches for concordance between alignments and transcripts. This approach might offer certain advantages compared to the alignment to transcriptome only, because it does not force the alignments to annotated transcripts.

By default, the output satisfies RSEM requirements: soft-clipping or indels are not allowed. `-- quantTranscriptomeBan Singleend` option allows insertions, deletions, and soft-clips in the transcriptomic alignments, which can be used by some expression quantification software (e.g., eXpress [4]).

3.3.4 Counting Number of Reads per Gene

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 [5] with default parameters. This option requires annotations (GTF or GFF with `--sjdbGTFfile` option) at the genome generation step or at the mapping step.

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 RNA strand (htseq-count option -s yes).
4. Counts for the stranded RNA-seq with the 2nd read strand matching the RNA strand (htseq-count option -s reverse).

Note that if you have stranded data and choose one of the columns 3 or 4, the other column (4 or 3) will give you the count of antisense reads.

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

References

1. Dobin A, Davis CA, Schlesinger F, Drenkow J, Zaleski C, Jha S, Batut P, Chaisson M, Gingeras TR (2013) STAR: ultrafast universal RNA-seq aligner. Bioinformatics 29:15–21
2. Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, Marth G, Abecasis G, Durbin R (2009) The sequence alignment/map format and SAMtools. Bioinformatics 25:2078–2079
3. Li B, Dewey CN (2011) RSEM: accurate transcript quantification from RNA-Seq data with or without a reference genome. BMC Bioinformatics 12:323
4. Roberts A, Pachter L (2012) Streaming fragment assignment for real-time analysis of sequencing experiments. Nat Methods 10:71–73
5. Anders S, Pyl PT, Huber W (2014) HTSeq—a Python framework to work with high-throughput sequencing data. Bioinformatics 31:166–169