RNA-Seq Workshop Manual

Chapter 3: Introduction to Alignment and Quantification: The STAR and featureCounts Tools

@ Alignment: The STAR Tool @

Goal: To align your reads to a reference genome.

Input files: Reference genome file (.fa) and gene annotation file (.gtf) to create an index; results from Trimmomatic (fastq.gz files) and the index files to align a sample
Output files: Aligned, sorted (by genomic coordinates) files (.bam); log files

Website: https://github.com/alexdobin/STAR
Manual: https://github.com/alexdobin/STAR/blob/master/doc/STARmanual.pdf

```
$ ssh your-username@hpc.brc.berkeley.edu
```

```
$ cd /global/scratch/users/your-username
```

Create a new conda environment and install STAR:

```
$ conda create -n star_env -c conda-forge -c bioconda star
```

Activate the Conda virtual environment:

```
$ conda activate star_env
```

Verify the installation:

```
$ STAR --version
```

Savio replies: 2.7.11b

Setting up STAR

There are three steps to set up STAR:

1) Obtaining the reference genome and annotation file, checking the two files for compatibility, and validating the two files using the checksums;
2) Building the index of the reference genome using STAR (instead of downloading it);
3) Running STAR (from either the command line or using a SLURM file).

Step 1. Obtain the mouse *reference genome* and the *gene annotation* file:

```
$ cd /global/scratch/users/your-username/
```

```
$ mkdir genome
```

```
$ cd genome
```

To see your current path:

```
$ pwd
```

(Should give: /global/scratch/users/your-username/genome)

Reference genome:

To ensure that the reference genome is compatible with the annotation set, download both of them from the same site:

Genome: GENCODE: FASTA (Primary Assembly, GRCm39, matched to vM36):

```
$ https://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_mouse/release_M36/
GRCm39.primary_assembly.genome.fa.gz (all one line)
```

Command to download the reference genome:
(Note: You are in the “genome” folder on Scratch)

```
$ wget https://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_mouse/release_M36/
GRCm39.primary_assembly.genome.fa.gz (all one line)
```

Download the gene annotation file:

```
$ wget https://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_mouse/release_M36/
gencode.vM36.chr_patch_hapl_scaff.annotation.gtf.gz (all one line)
```

Check Compatibility Once Downloaded:

A. Unzip the files:

```
$ gunzip -c GRCm39.primary_assembly.genome.fa.gz | grep "^>" | cut -d " " -f 1 | sed 's/>//' |
sort | uniq > gencode_vM36_genome_chr_names.txt (all one line)
```

```
$ gunzip -c gencode.vM36.chr_patch_hapl_scaff.annotation.gtf.gz | awk '$1 !~ /^#/' | cut -f1 |
sort | uniq > gencode_vM36_annotation_chr_names.txt (all one line)
```

B. Compare the two files: `comm -3` shows you the **differences** between the two files, ignoring their common lines:

```
$ comm -3 gencode_vM36_genome_chr_names.txt gencode_vM36_annotation_chr_names.txt
```

Savio replies:

```
GL456394.1
GL456396.1
JH584295.1
JH584300.1
JH584301.1
JH584302.1
MU069434.1
MU069435.1
```

Note:
From NCBI's documentation: Contigs with names like GL..., JH..., MU... represent unlocalized or unplaced scaffolds. So, your files are compatible. If your GTF annotation file uses

chromosome names that do not exist in the genome FASTA, then those names will appear only in the GTF, and will show up in your comm -3 output. Suppose instead your comm -3 output looks like this:

```
GL456233.2
GL456354.1
GL456385.1
chr1
chr2
chr3
chrX
chrY
```

This means that your annotation file uses chr1, chr2, etc., but those terms are **missing** from your reference genome FASTA file (you may additionally see: 1, 2, . in addition to chr1, chr2, … if the files are not compatible). Thus, in this example, you have incompatibility.

Verify that your files are not corrupted using CHECKSUMS:

How to get the Checksum file for all files:

```
$ wget https://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_mouse/release_M36/
MD5SUMS
```

a) Reference Genome file:

Look inside the checksums file:

```
$ cat MD5SUMS
```

Looking inside the checksum file, we find:

```
c7be5db71681df7e457f22f0375e65c8  GRCm39.primary_assembly.genome.fa.gz
```

Run the bash command:

```
$ md5sum GRCm39.primary_assembly.genome.fa.gz
```

Output:

```
c7be5db71681df7e457f22f0375e65c8  GRCm39.primary_assembly.genome.fa.gz
```

Yes, they match, so no file corruption from the download for the reference genome.

b) Annotation file:

Next, let's check the annotation file:
The annotation file is: gencode.vM36.chr_patch_hapl_scaff.annotation.gtf.gz

Looking inside the checksum file, we find the annotation file checksum:

```
d7fe1f701c5d9998f3d0e903b0868d69  gencode.vM36.chr_patch_hapl_scaff.annotation.gtf.gz
```

Run the bash command:

```
$ md5sum gencode.vM36.chr_patch_hapl_scaff.annotation.gtf.gz
```

Output:

```
d7fe1f701c5d9998f3d0e903b0868d69  gencode.vM36.chr_patch_hapl_scaff.annotation.gtf.gz
```

They match, so the annotation file is not corrupted.

Step 2: Build index of reference genome for STAR
Data needed: A reference genome (FASTA file) and a gene annotation file (GTF file):

GRCm39.primary_assembly.genome.fa.gz (gunzipped FASTA file)

gencode.vM36.chr_patch_hapl_scaff.annotation.gtf.gz (gunzipped GTF file)

Rename these two files:

```
$ cp GRCm39.primary_assembly.genome.fa.gz genome.fa.gz
$ cp gencode.vM36.chr_patch_hapl_scaff.annotation.gtf.gz annotation.gtf.gz
```

Unzip these gunzipped files:

```
$ gunzip genome.fa.gz

$ gunzip annotation.gtf.gz
```

Your location: /global/scratch/users/your-username/genome

```
$ mkdir index_star
```

Generic Code to create index:

```
STAR \
--runMode genomeGenerate \
--genomeDir /path/to/index_star \  (output files)
--genomeFastaFiles /path/to/genome.fa \  (reference genome - input file)
--sjdbGTFfile /path/to/annotation.gtf \ (genome annotation - input file)
--runThreadN -1 (run all available cores; note: no "\" here)
```

Example code to create index:

```
$ STAR \
> --runMode genomeGenerate \
> --genomeDir index_star \
> --genomeFastaFiles genome.fa \
> --sjdbGTFfile annotation.gtf \
> --runThreadN 12 (Using 12 threads for faster processing)
$
```

Savio replies:

```
10:38:45 ..... started STAR run
12:26:04 ..... finished successfully
```

Step 3: Run STAR: Alignment

(mapping paired-end reads to reference genome)

```
$ cd /global/scratch/users/your-username

$ cd mouse_data

$ mkdir star_results

$ cd /global/scratch/users/your-username
```

Option A. Run STAR on the command line (for paired-ended reads):

Generic Code for running STAR on a single sample:
(You're in the Scratch base directory)

```
$ STAR \
--runMode alignReads \
--genomeDir /path/to/index_star \
--readFilesIn /path/to/sample_1_R1.fastq.gz /path/to/sample_1_R2.fastq.gz \
--outFileNamePrefix /desired/output/dir/sample_1_ \  (Specifies the prefix for the output files;
the output files will start with sample_1_)
--outSAMType BAM SortedByCoordinate \ (BAM is the compressed version of SAM)
--readFilesCommand 'gunzip -c' \ (to unzip/decompress the input FASTQ files)
--runThreadN 12    (Using 12 threads for faster processing; no "\" here since it's the last line)
```

Example code for running STAR on a single sample:
(You're in the Scratch base directory)

```
$ STAR \
> --runMode alignReads \
> --genomeDir genome/index_star \
> --readFilesIn mouse_data/trim_results/ctrl_rep1_1_paired.fastq.gz mouse_data/trim_results/
ctrl_rep1_2_paired.fastq.gz \ (all one line)
> --outFileNamePrefix mouse_data/star_results/ctrl_rep1_ \
> --outSAMtype BAM SortedByCoordinate \
> --readFilesCommand 'gunzip -c' \
> --runThreadN 12
```

Savio replies:

```
13:12:37 ..... started STAR run
13:16:41 ..... finished successfully
```

Output file has name: ctrl_rep1_Aligned.sortedByCoord.out.bam

We want to rename the files to make our life simpler, so use the "cp" command or the "mv" command):

Example:

```
$ mkdir renamed

$ cp ctrl_rep1_Aligned.sortedByCoord.out.bam ctrl_rep1.bam
$ mv ctrl_rep1.bam renamed
```

So, now all your cleaned-up (renamed) STAR BAM files are located in another place, the “renamed” folder.

Option B. Use the SLURM file to align a single sample:

Go on the workshop website and download the SLURM file for STAR, namely, mouse_star.sh: (“$” and “>” are cursors so you don’t type them; be sure to type the “\”):

```
$ wget -O mouse_star.sh \
> https://raw.githubusercontent.com/elinorv21/RNA-Seq_workshop/main/mouse_star.sh
```

Then edit the file as appropriate:

```
#SBATCH --job-name=mouse_star_sample1
#SBATCH --output=mouse_star_%j.out
#SBATCH --error=mouse_star_%j.err
#SBATCH --time=01:00:00

GENOME_DIR=genome/index_star  # Path to STAR index from scratch base directory
INPUT_R1=mouse_data/trim_results/trim_paired_sample1_1.fastq.gz # _1 = forward
INPUT_R2=mouse_data/trim_results/trim_paired_sample1_2.fastq.gz # _2 = reverse (note: we are using relative path not absolute path)
OUTPUT_DIR=mouse_data/star_results
SAMPLE_NAME=sample1  # For output file naming
```

Finally, run the job:

```
sbatch mouse_star.sh
```

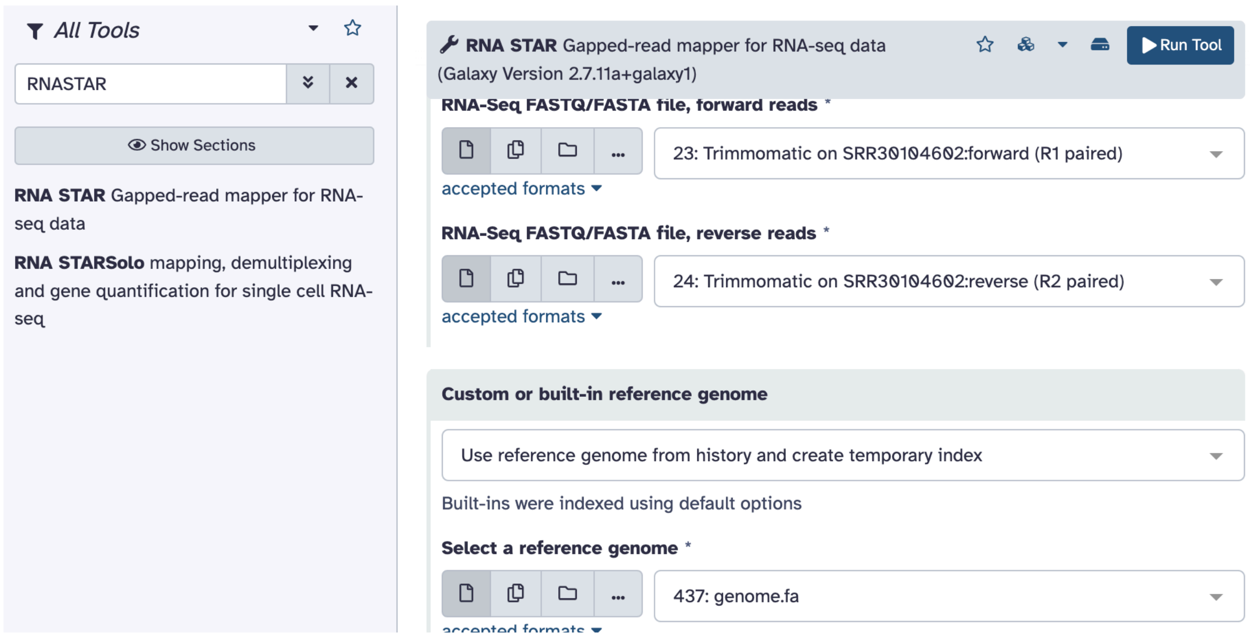
Savio will reply with your jobID number. Use this jobID to identify your appropriate .err and .out files.

Add this code to the SLURM file if you want Savio to send you an email when the SLURM job is completed:

```
#SBATCH --mail-user=your.email@domain.com
#SBATCH --mail-type=END,FAIL
#SBATCH --mail-from=no-reply@slurm.cluster   # Optional
```

Option C. Use STAR on Galaxy

1. Search for ‘RNA STAR’ in Tools
2. Select the Trimmomatic output or upload the necessary files (trimmed fastq files)
A) Paired or Unpaired (assuming paired here)
B) Upload the reference genome for mouse (.fa file) to create a temporary index (Galaxy makes the index)
3. See running STAR on Savio for details about the options.



RNA STAR Gapped-read mapper for RNA-seq data

RNA STARSolo mapping, demultiplexing and gene quantification for single cell RNA-seq

RNA STAR Gapped-read mapper for RNA-seq data (Galaxy Version 2.7.11a+galaxy1)

RNA-Seq FASTQ/FASTA file, forward reads *
23: Trimmomatic on SRR30104602:forward (R1 paired)

RNA-Seq FASTQ/FASTA file, reverse reads *
24: Trimmomatic on SRR30104602:reverse (R2 paired)

Custom or built-in reference genome
Use reference genome from history and create temporary index
Built-ins were indexed using default options

Select a reference genome *
437: genome.fa

Figure 1. Using Galaxy to run STAR

@ Quantification: FeatureCounts: @

Goal: Generate a count_matrix for analysis by DESeq2, edgeR, limma. Alternative splicing analysis uses the STAR output (.bam files).

Website and Manual: https://subread.sourceforge.net/featureCounts.html

Installation:
(You are in the Scratch base directory)
Conda installation:

Create a new conda environment and install featureCounts via subread:
```
$ conda create -n feature_env -c conda-forge -c bioconda subread
```

Activate the Conda virtual environment:
```
$ conda activate feature_env
```

To test installation:
```
$ featureCounts -v
```

Savio replies:
```
featureCounts v2.1.1
```

Make a folder for featureCounts output:
```
$ cd mouse_data
$ mkdir fcounts_results
```

```
$ cd /global/scratch/users/your-username
```

Run featureCounts:

Generic code:

(You are in the Scratch base directory)

```
featureCounts \
-a path/to/annotation.gtf \
-o /path/to/output/counts.txt \
-T 8 \ (8 threads)
-t exon \
-g gene_id \
-p -B -C \
--primary \
--countReadPairs \
path/to/file/sample1_baseline.bam path/to/file/sample2_baseline.bam path/to/file/
sample3_baseline.bam path/to/file/sample1_treatment.bam path/to/file/path/to/file/
sample2_treatment.bam path/to/file/sample3_treatment.bam (all one line)
Or,
path/to/files/*.bam
```

Description of the options:

-a: annotation.gtf file (include the path so the software can find the gtf file)
-o: the name of the output file needs to be specified; include the desired path
-T: the number of processors/threads to be used for this calculation
-t: specify “exon” or “gene” (specify “exon” for gencode annotation file)
-g: specify the name of the gene IDs, “gene_id” in this case
-p: need this option for paired ends
-B: having this option requires both ends of a paired read to be successfully aligned
-C: having this option avoids counting chimeric fragments, where paired reads align to different chromosomes
--primary: Only count primary alignments (useful when dealing with multi-mapped reads)
--countReadPairs: need for paired reads

Example code for running featureCounts on all samples:
(You’re in the Scratch base directory: Typing “pwd” gives /global/scratch/users/your-username)

```
$ featureCounts \
> -a genome/annotation.gtf \
> -o mouse_data/fcounts_results/gene_counts.txt \
> -t exon -g gene_id \
> -p -B -C \
> --primary --countReadPairs \
> mouse_data/star_results/renamed/*.bam
```

Output files: gene_counts.txt, gene_counts.txt.summary

Alternative: Use a SLURM file to run featureCounts on your data:

Go on the workshop website and download the SLURM file for featureCounts, namely, mouse_fcounts.sh:
(“$” and “>” are cursors so you don’t type them; be sure to type the “\”):

```
$ wget -O mouse_fcounts.sh \
> https://raw.githubusercontent.com/elinorv21/RNA-Seq_workshop/main/mouse_fcounts.sh
```

Then edit the file as appropriate:

```
#SBATCH --job-name=fcounts
#SBATCH --output=fcounts_%j.out
#SBATCH --error=fcounts_%j.err
#SBATCH --time=01:00:00

BAM_FILES=(
# fill in here the name and location of your files
)

GTF_FILE=genome/annotation.gtf
OUTPUT_FILE=mouse_data/fcounts_results/gene_counts.txt
```

Finally, run the job:

```
sbatch mouse_fcounts.sh
```

Savio will reply with your jobID number. Use this jobID to identify your appropriate .err and .out files.

Notes:

1. Do not use featureCounts on Galaxy.

2. Interpretation: To get an understanding of the results:

$ conda install samtools (you’re in the feature_env environment)

$ samtools --version (Savio replies with version number and other information)

```
$ samtools flagstat /global/scratch/users/your-username/aligned_data_folder/sample.bam
```

Savio replies (**example**):

“38998680 + 0 properly paired (100.00% : N/A)” (properly paired reads), so
19499340 valid read pairs (properly paired, aligned fragments).

Then, after the featureCounts command, use:

```
$ awk 'NR>2 {sum += $7} END {print sum}' gene_counts.txt
```

Savio replies:

14716210 (fragments (assigned to genes) - counted by featureCount)

Thus, ~75% of the high-quality fragments in the BAM file were successfully counted toward at least one gene (14716210/19499340 = 75%).