# DSIB01 Autumn 2021

## 04 Alignment

Mgr. Eliška Chalupová
375973@mail.muni.cz

# Practicals overview

- STAR
- Repetitive elements removal
- Reads alignment
- Deduplication
- SAM file format & samtools

# Preliminary quality check

- Not necessary if you did one at the end of the last practicals

- Use loops when running the same command for multiple files

```
for file in /home/user/encode/*.fastq
 do
   fastqc -o /home/user/encode/fastqc/01 $file
 Done
```

- Use multiQC to compare multiple results

```
conda install -c bioconda multiqc
```

```
multiqc /home/user/fastqc/01 -o /home/user/multiqc/01
```

# Technical tips

- Environment variables
  - There are no spaces when defining environment variables
  - Use '$' sign to reference defined variables
  - You can manipulate them through ${READ1}
  - More information e.g. here

```
READ1=ENCFF708YAL
READ2=ENCFF959XKN
OUT_DIR=/home/user/output
```

```
echo $READ1
echo $READ2
```

```
echo ${READ1#ENCFF}
echo ${READ2#ENCFF}
```

- When specifying an output directory, first make sure it exists
  - Use option -p to create multiple nested directories at the same time

```
mkdir -p /home/user/output/star/repeats
```

# Repositioning UMIs

- UMIs will be explained later in this practicals, for now we just need to do one step before alignment
- Check the length of UMIs in your fastq files and adjust the length (l=10) parameter accordingly
  - Use the head command `head -n 4 /home/user/output/cutadapt/round2/$READ1.adapterTrim.fastq`

```
@NAGGCCCTAC:K00180:234:HCTHMBBXX:4:1101:26494:1015 2:N:0:NAATTCGT+NTTCGCCT
GTTTCGCGGCCAAGATGGTGCTGGTGCTGCGCCATCCTTT
+
JJJFJJJJJJJJJJJJJJJJJJJJJJJJJJJJAJJJJJ
```

- Run the following command for both fastq files to reposition UMIs

```
awk -v l=10 'BEGIN{OFS=FS=" "} substr($1, 1, 1) == "@" {print "@" substr($1, (l+3), 500) "_"
substr($1, 2, l) }; substr($1, 1, 1) != "@" {print}; '
/home/user/output/cutadapt/round2/$READ1.adapterTrim.fastq >
/home/user/output/umis/$READ1.adapterTrim.umi.fastq
```

# Alignment

- There are multiple alignment tools available
- Each tool has many parameters with many options
- The choice of the tool and parameters is crucial

- We have to understand our data to make the right choices
- We have to understand the tool and its options

- E.g., if we do not allow any mismatches, it is not possible to detect SNPs. If we allow too many mismatches, we get too many false SNPs and wrong alignments.

# STAR

- Installation

```
conda activate Environment
conda install -c bioconda star
```

- Manual

  https://physiology.med.cornell.edu/faculty/skrabanek/lab/angsd/lecture_notes/STARmanual.pdf

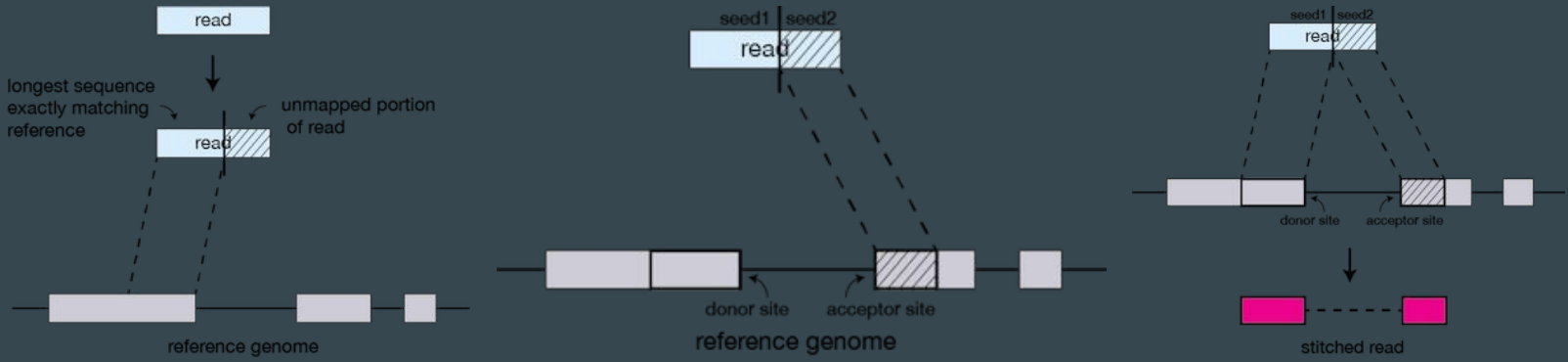
- Original paper at https://pubmed.ncbi.nlm.nih.gov/23104886/

# STAR - Indexing

- First generate genome index, unless available, by running STAR in runMode 'genomeGenerate'
- Genome files comprise binary genome sequence, suffix arrays, text chromosome names/lengths, splice junctions coordinates, and transcripts/genes information. Most of these files use internal STAR format and are not intended to be utilized by the end user.
- --sjdbGTFfile - Optional, STAR will extract splice junctions from the GTF file and use them to improve accuracy of the mapping.
- --genomeSAindexNbases - For small genomes, this parameter must be scaled down, with a typical value of min(14, log2(GenomeLength)/2 - 1). For example, for 1 megaBase genome, this is equal to 9, for 100 kiloBase genome, this is equal to 7.
- --genomeChrBinNbits - If you are using a genome with a large number of references (>5,000 chromosomes/scaffolds), you may need to lower this parameter to reduce RAM consumption. For example, for 3 gigaBase genome with 100,000 chromosomes/scaffolds, this is equal to 15.

# STAR - Alignment

- STAR (Spliced Transcripts Alignment to a Reference) is an aligner designed to specifically address many of the challenges of RNA data mapping by accounting for spliced alignments
- Outperforms other aligners in mapping speed, but it is memory intensive
- The algorithm achieves this highly efficient mapping by performing a two-step process: 1) Seed searching, and 2) Clustering, stitching, and scoring




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

# STAR - Alignment

- Note that “STAR’s default parameters are optimized for mammalian genomes. Other species may require significant modifications of some parameters; in particular, the maximum and minimum intron sizes have to be reduced for organisms with smaller introns”.
- --outFilterMultimapNmax - Default filtering allows maximum of 10 multiple alignments for a read. If it is exceeded, no alignment is outputted.
- The logs and other output files are created by STAR at the current working directory by default - make sure to be at the right place `cd /home/user/output/star/repeats` or use option --outFileNamePrefix /home/user/output/star/repeats/

# STAR - ENCODE options

- --outFilterType BySJout - reduces the number of “spurious” junctions
- --outFilterMultimapNmax 20 - max number of multiple alignments allowed for a read: if exceeded, the read is considered unmapped
- --alignSJoverhangMin 8 - minimum overhang for unannotated junctions
- --alignSJDBoverhangMin 1 - minimum overhang for annotated junctions
- --outFilterMismatchNmax 999 - maximum number of mismatches per pair, large number switches off this filter
- --outFilterMismatchNoverReadLmax 0.04 - max number of mismatches per pair relative to read length: for 2x100b, max number of mismatches is 0.04*200=8 for the paired read
- --alignIntronMin 20 - minimum intron length
- --outSAMunmapped Within - output unmapped reads within the main SAM file
- --alignEndsType EndToEnd - In eCLIP the cross linking position should be at the beginning of the second read. If we would enable soft-clipping, we would add potential bases with low quality at the end of our second reads that would blur our cross linking position.

# Repetitive elements - RepBase

- “A substantial portion of eukaryotic genomes is composed of multiple DNA copies referred to as “repetitive DNA”, which can be divided into two major groups” - tandem repeats and transposable (selfish) elements
- Over 40% of the human genome is still composed of recognizable interspersed repeats of which some are over 200 million years old

  Read more at Repbase Update, a database of eukaryotic repetitive elements

- Recommendation from eCLIP-seq Processing Pipeline - “Removing repetitive elements helps control for spurious artifacts from rRNA (and other) repetitive reads”
- Case against filtering out the repetitive elements - “By focusing on only a fraction of the genome, only a fraction of discoveries can be made.”

# Repetitive elements

1. Generate index - apply the option for small genomes

```
STAR \
--runMode genomeGenerate \
--genomeSAindexNbases 5 \
--runThreadN 2 \
--genomeDir /home/user/ref/repeats \
--genomeFastaFiles /home/user/ref/repeats/RepBase_hs_shared_11272018.fasta
```

# Repetitive elements

2. Align the reads

```
STAR --runThreadN 2 \
--genomeDir /home/user/ref/repeats \
--readFilesIn home/user/output/cutadapt/round2/ ${READ1%.fastq}.adapterTrim.fastq \
home/user/output/cutadapt/round2/ ${READ1%.fastq}.adapterTrim.fastq \  \
-outSAMunmapped Within \
--outSAMattributes All \
--outStd BAM_Unsorted \
--outSAMtype BAM SortedByCoordinate \
--outFilterType BySJout \
--outReadsUnmapped Fastx \
--outFileNamePrefix /home/user/output/star/repeats/ \
--alignEndsType EndToEnd
```

# Repetitive elements

- Reads corresponding to the repetitive elements got aligned
- However, we are interested in those, that did not align
- Further on, we will be working with the unmapped files Unmapped.out.mate1 and Unmapped.out.mate2 - those are the reads with the repetitive elements removed
- We can give them more meaningful names, e.g.

```
cd /home/user/output/star/repeats/
mv Unmapped.out.mate1 $READ1.rm_rep.fastq
mv Unmapped.out.mate2 $READ2.rm_rep.fastq
```

# Reads alignment

- We will align against the newest human genome assembly - Hg38
  - We will use only chromosome 1 for the purposes of this practicals
  - Using only chromosome 1 we need to lower the parameter --genomeSAindexNbases to 12
  - You can get all the genome files e.g. at USCS https://hgdownload.soe.ucsc.edu/goldenPath/hg38/bigZips/
- First, we need to index the genome again
  - For the purposes of the practicals, the prepared indexed files were sent to your email

```
STAR --runMode genomeGenerate \
--runThreadN 2 \
--genomeSAindexNbases 12 \
--genomeDir /home/user/ref/chr1 \
--genomeFastaFiles /home/user/ref/chr1/chr1.fa
```

# Reads alignment

```
STAR --runThreadN 2 \
--genomeDir /home/user/ref/chr1 \
--readFilesIn /home/user/output/star/repeats/$READ1.rm_rep.fastq \
/home/user/output/star/repeats/$READ2.rm_rep.fastq \
--outSAMunmapped Within \
--outFilterMultimapNmax 20 \
--alignSJoverhangMin 8 \
--alignSJDBoverhangMin 1 \
--outFilterMismatchNmax 99 \
--outFilterMismatchNoverReadLmax 0.04 \
--alignIntronMin 20 \
--outSAMattributes All \
--outSAMtype BAM SortedByCoordinate \
--outFilterType BySJout \
--outReadsUnmapped Fastx \
--outFileNamePrefix /home/user/output/star/chr1/ \
--alignEndsType EndToEnd
```

# STAR - Output

- Aligned.out.sam / Aligned.out.bam / Aligned.sortedByCoord.out.bam - alignments in standard SAM/BAM format
- Log.out - main log file with a lot of detailed information about the run
- Log.progress.out - reports job progress statistics, such as the number of processed reads
- Log.final.out - summary mapping statistics, useful for quality control
- SJ.out.tab - contains high confidence collapsed splice junctions in tab-delimited format
- Unmapped.out.mate1 & Unmapped.out.mate1 - unmapped reads in original fastq format (thanks to option --outReadsUnmapped set to Fastx)

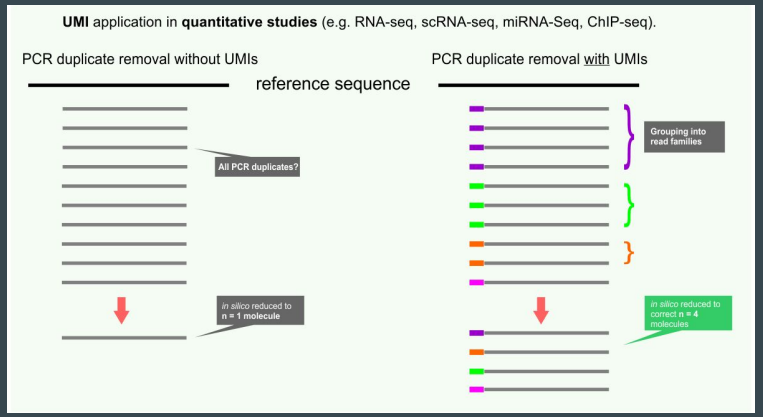
Images taken from: https://hbctraining.github.io/Intro-to-rnaseq-hpc-O2/lessons/03_alignment.html

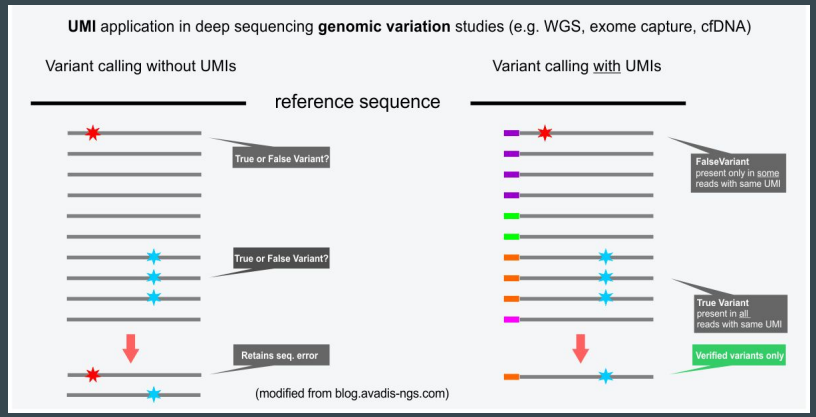
# Deduplication

- PCR duplicates are reads that are made from the same original cDNA molecule via PCR.
- A common practice to eliminate PCR duplicates is to remove all but one read of identical sequences.
- For example, a large number of PCR duplicates containing an amplification-induced error may cause a variant calling algorithm to misidentify the error as a true variant.
- However, several studies have shown that retaining PCR- and Illumina clustering duplicates does not cause significant artifacts as long as the library complexity is sufficient (e.g. here).
- PCR duplicates are thus mostly a problem for very low input or for extremely deep RNA-sequencing projects.

# Deduplication - UMIs

- UMIs (Unique Molecular Identifiers) should be used to prevent the removal of natural duplicates.
- UMIs, or molecular barcodes, are short sequences used to uniquely tag each molecule in a sample library.
- UMIs are added before PCR amplification, and can be used to reduce errors and quantitative bias introduced by the amplification.



# Deduplication - UMI-tools

- UMI-tools contains tools for dealing with Unique Molecular Identifiers (UMIs)/Random Molecular Tags (RMTs) and single cell RNA-Seq cell barcodes.
- Installation

```
conda create -n umi python=3.7 # UMI-tools does not work with the latest python version
conda activate umi
conda install -c bioconda -c conda-forge umi_tools
```

- Usage

  --dedup - Use this when you want to remove the PCR duplicates

  --group - This is useful when you want to manually interrogate the PCR duplicates or perform bespoke downstream processing such as generating consensus sequences

  --count - Use this when you want to obtain a matrix with unique molecules per gene, per cell, for scRNA-Seq

# Deduplication

- UMI-tools require the input BAM file to be indexed
- To do that, we will use samtools

```
samtools index /home/user/output/star/hg38_chr1/Aligned.sortedByCoord.out.bam
```

- Now we do the deduplication

```
umi_tools dedup --stdin=/home/user/output/star/hg38_chr1/Aligned.sortedByCoord.out.bam \
--log=/home/user/output/dedup/chr1_dedup.log  \
> /home/user/output/dedup/chr1_dedup.bam
```

- We can check the results of the deduplication in the log file
- The deduplicated file (chr1_dedup.bam) will be used for the following steps

# Sequence Alignment Map (SAM)

- It is a TAB-delimited text format consisting of a header and an alignment section
- The alignment section contains the information for each sequence about where/how it aligns to the reference genome
- Each alignment line has 11 mandatory fields for essential alignment information such as mapping position, and variable number of optional fields for flexible or aligner specific information.
- BAM - binary version, compressed, not human-readable, required by some tools for downstream analysis
- See more information at https://genome.sph.umich.edu/wiki/SAM or http://samtools.github.io/hts-specs/SAMv1.pdf

# Sequence Alignment Map (SAM)

- Each alignment has:
  - query name - used to group/identify alignments that are together, like paired alignments or a read that appears in multiple alignments
  - a bitwise set of information describing the alignment, FLAG. Provides the following information:
    - are there multiple fragments?
    - are all fragments properly aligned?
    - is this fragment unmapped?
    - is the next fragment unmapped?
    - is this query the reverse strand?
    - is the next fragment the reverse strand?
    - is this the 1st fragment?
    - is this the last fragment?
    - is this a secondary alignment?
    - did this read fail quality controls?
    - is this read a PCR or optical duplicate?

# Samtools

- Set of utilities that manipulate alignments in the SAM, BAM, and CRAM formats
- Converts between the formats, does sorting, merging and indexing, and can retrieve reads in any regions swiftly
- Documentation http://www.htslib.org/doc/samtools.html
- Installation `conda install -c bioconda samtools`
- index - index a sorted SAM or BAM file for fast random access
- flagstat - calculates statistics based primarily on the bit flags (see the flags description here)
- view - with no options or regions specified, prints all alignments in the specified input alignment file to the stdout in the SAM format. Use of region specifications requires a coordinate-sorted and indexed input file.

# Finalization

- We can look how at the first alignment as an example of SAM format

```
samtools view /home/user/output/dedup/chr1_dedup.bam |  head -n 1
```

- Index the deduplicated file

```
samtools index /home/user/output/dedup/chr1_dedup.bam
```

- Calculate statistics

```
samtools flagstat /home/user/output/dedup/chr1_dedup.bam
```

- You can save them to the file using ‘>’ sign

```
samtools flagstat /home/user/output/dedup/chr1_dedup.bam \
> /home/user/output/dedup/chr1_dedup.bam.flagstat
```

# Project task

1. Map both read files to the repetitive elements
2. Use the unmapped files to map them to the chromosome 1 of human genome
3. Perform deduplication of the mapped reads
4. Perform quality check of aligned deduplicated bam file
5. Get statistics about the deduplicated file using samtools

- Mark and discuss all the results in your project report
- Push the Alignment.sh script to Your GitHub repository