RNAseq pipeline

Planning and notes

Step 1: Index reference genome with transcript annotations

- Decided to use STAR
  - Pros: Gold-standard for a long time, lots of tutorials. Performs a full alignment making downstream analysis more customizable.
  - Cons: not the fastest algorithm for just counting reads that align to transcripts.
  - Other options include Salmon/Kallisto (K-mer based methods).
    - These are faster, but perform more of a pseudoalignment. More suitable for doing analyses that require looking at multiple gene isoforms.
- Reference genome:
  - GRCm39 primary assembly = core set of chromosomes, without extra scaffolds and isoforms (19 autosomes and 2 sex chromosomes).
  - Paired with GencodeM36 primary assembly annotation (.gtf file).
  - These must be unzipped using gunzip before running the indexing.

Reminder on job submission with SLURM:
To submit:
sbatch JOBNAME.sh
To check:
squeue –u $USER
To cancel:
scancel <jobid>

- Job Script:

Index.sh

```
#!/bin/bash
#SBATCH --job-name=mlh1STARindex.sh
#SBATCH -n 1
#SBATCH -N 1-1
#SBATCH -p all          #partition name
#SBATCH -c 12           #number of cores requested, must be
greater or equal to the number needed for the job
#SBATCH --mem=64G               #memory
#SBATCH -t 0-2:00               #Hours:minutes runlimit after
which the job will be killed
#SBATCH -o %j.out               #std out file
#SBATCH -e %j.err               #std err file

module load Python/3.6.5
module load FastQC
module load STAR

STAR --runThreadN 12 \
--runMode genomeGenerate \
--genomeDir mm39_index \
--genomeFastaFiles
/home/bosia/RNAseq/GencodeM36/GRCm39.primary_assembly.genome.fa \
--sjdbGTFfile
/home/bosia/RNAseq/GencodeM36/gencode.vM36.primary_assembly.annota
tion.gtf \
--sjdbOverhang 149
```

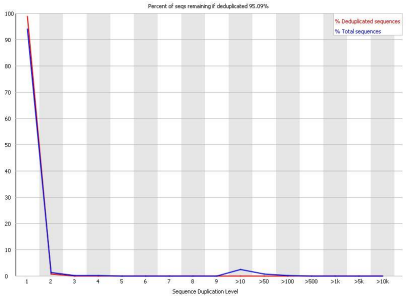
Step 2: Check sequencing quality using FastQC

- Decided to do this step because there were some questions about the quality of the RNA we sent for sequencing. It is technically optional, however.
- Checking data by FastQC tells a few important things:
  - Overall quality of the sequencing
    - Looks great in this case, all bases have high quality scores.
  - Does duplicate sequence frequency show something might be off?
    - For RNAseq, it is normal to see a slight spike around >10 to >100 sequence duplication level. This is true for other types of enriched experiments. Sequences left after de-duplication is usually around 40-50% for RNAseq experiments.
  - Is there adapter contamination?
    - FastQC shows mild adapter contamination at the 5' ends of sequences with the universal illumina adapter. We should remove these (next slide)
- Use scp to move outputs to sequencing folder on network. Can view html reports for each sequencing file.

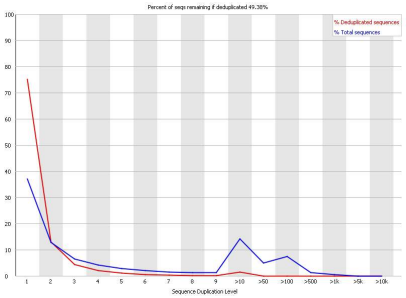
- Job Script:

```
#!/bin/bash
#SBATCH --job-name=mlh1STARmap.sh
#SBATCH -n 1
#SBATCH -N 1-1
#SBATCH -p all          #partition name
#SBATCH -c 12           #number of cores requested, must be
greater or equal to the number needed for the job
#SBATCH --mem=64G               #memory
#SBATCH -t 0-2:00               #Hours:minutes runlimit after
which the job will be killed
#SBATCH -o %j.out               #std out file
#SBATCH -e %j.err               #std err file

module load Python/3.6.5
module load FastQC
module load STAR

#preprocess all reads files with FASTQC
fastqc -t 12 /home/bosia/RNAseq/Genewiz_data/30-
1121098857/00_fastq/*

#move files to new folder
mv *fastqc* /home/bosia/RNAseq/Genewiz_data/FQC_results
```

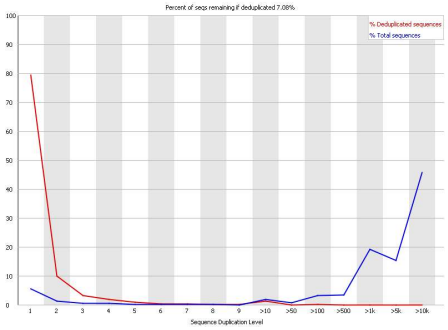
Example of very diverse library (i.e. WGS)



Example of normal RNAseq duplication levels

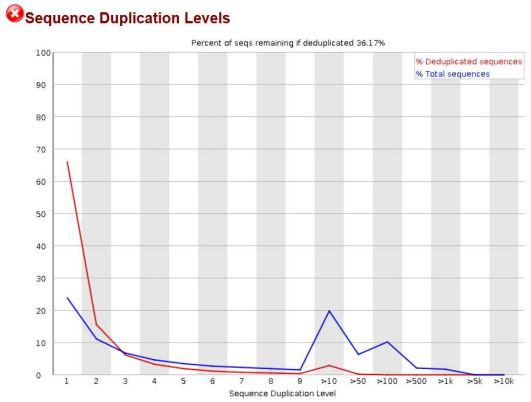


Example of PCR duplicate contaminated sample

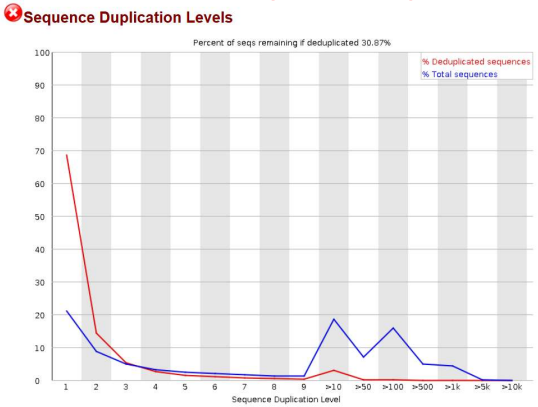


Sequence duplication levels give some sense of sequence diversity in the sample

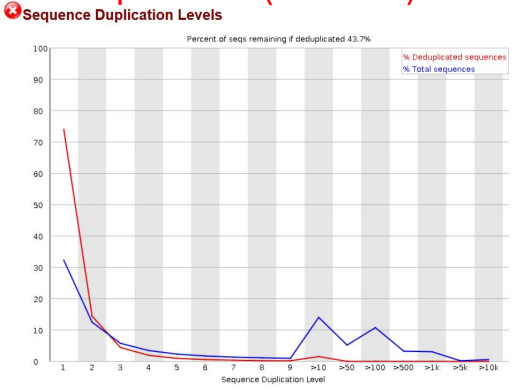
Sample: R13 (WT D10)



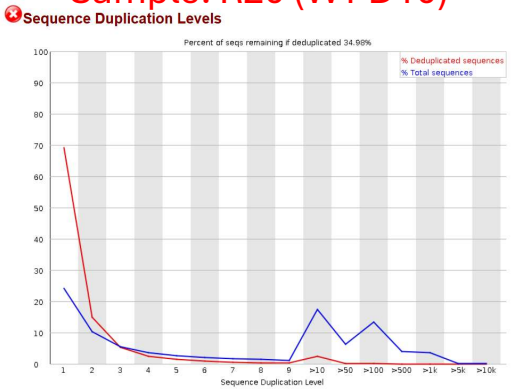
Sample: R16 (WT D10)



Sample: R18 (WT D10)



Sample: R20 (WT D10)



Example graphs from: https://proteo.me.uk/2013/09/a-new-way-to-look-at-duplication-in-fastqc-v0-11/

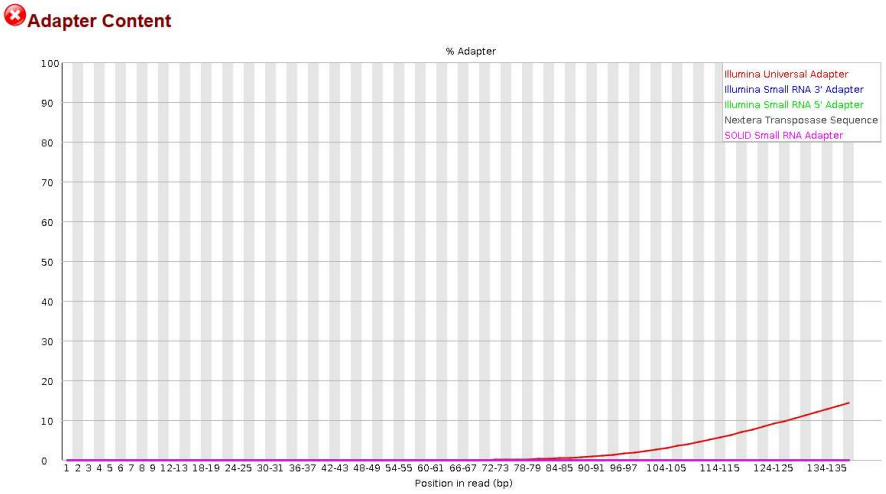
Step 3: Remove adapter contamination

- Use CutAdapt to remove the readthrough of Illumina universal adapters on 3' ends of forward and reverse sequences.
- Use the paired-end mode for paired-end sequencing for more accurate trimming.
- If you run FastQC again on trimmed reads files, the adapter spike should disappear completely, but your reads will not be all one length (150bp) and will "fail" the length assessment. This is normal.
- I batched 3 sets of reads files into 4 separate job scripts to process all 12 samples. This allows me to trim 4 reads files in parallel, while each job script will trim adapters on 3 samples sequencially.
  - This makes the whole process take about 40 minutes (instead of a few hours).
  - An array job may also be appropriate for this type of action.
  - Be sure to set the "-m 50" option for cutadapt, or there will be blank reads that cause problems in the alignment step.

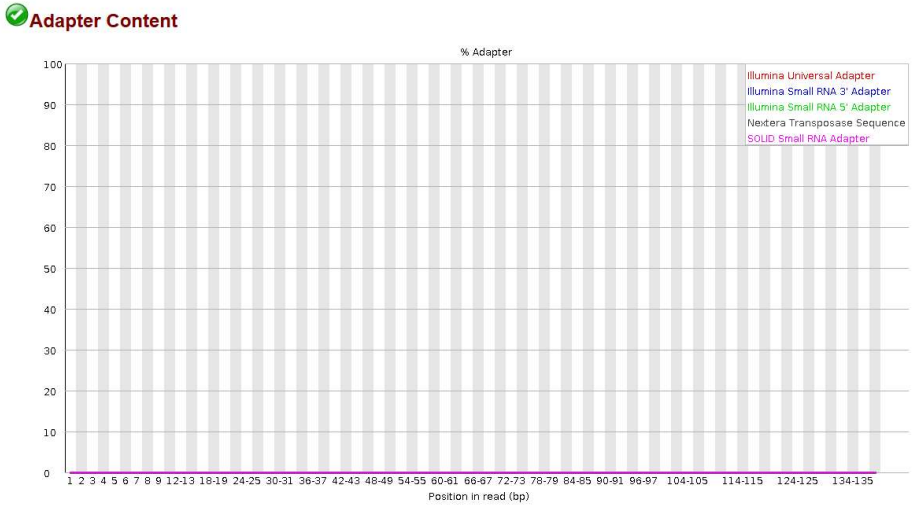
- Job Script:

```
#!/bin/bash
#SBATCH --job-name=mlh1STARmap.sh
#SBATCH -n 1
#SBATCH -N 1-1
#SBATCH -p all          #partition name
#SBATCH -c 12           #number of cores requested, must be
greater or equal to the number needed for the job
#SBATCH --mem=16G               #memory
#SBATCH -t 0-2:00               #Hours:minutes runlimit after
which the job will be killed
#SBATCH -o %j.out               #std out file
#SBATCH -e %j.err               #std err file

module load Python/3.6.5
module load cutadapt
module load pigz
module load STAR

cd /home/bosia/RNAseq/Genewiz_data/30-1121098857/00_fastq

#Trim adapters using cutadapt (FastQC results showed illumina
universal adapter contamination on the 5' ends of both reads
files)
cutadapt -j 12 -a AGATCGGAAGAG -A AGATCGGAAGAG –m 50 -o
tr_R4_R1_001.fastq.gz -p tr_R4_R2_001.fastq.gz Mlh1ko-d10-
R4_R1_001.fastq.gz Mlh1ko-d10-R4_R2_001.fastq.gz

cutadapt -j 12 -a AGATCGGAAGAG -A AGATCGGAAGAG –m 50 -o
tr_R7_R1_001.fastq.gz -p tr_R7_R2_001.fastq.gz Mlh1ko-d24-
R7_R1_001.fastq.gz Mlh1ko-d24-R7_R2_001.fastq.gz

cutadapt -j 12 -a AGATCGGAAGAG -A AGATCGGAAGAG –m 50 -o
tr_R9_R1_001.fastq.gz -p tr_R9_R2_001.fastq.gz Mlh1ko-d24-
R9_R1_001.fastq.gz Mlh1ko-d24-R9_R2_001.fastq.gz
```

Example of adapter removal results:



Sample R5_R1 before cutadapt:
3’ end contamination with Illumina universal adapter sequence
(This results from adapter read-through and is very common in NGS)



Sample R5_R1 After cutadapt:
3’ end contamination with Illumina universal adapter sequence
(This results from adapter read-through and is very common in NGS)

Step 4: Aligning reads to indexed/annotated reference genome.

- Alignment with STAR – see job script for parameters.
- IF successful, should generate many files.
  - Sample_Log.final.out files show basic alignment outputs and stats (table below shows some parameters).
  - Alignments are all pretty good (>85% of reads were mapped).

Mapping Data	MLH1-KO D10			MLH1-KO D24			MLH1-WT D10			MLH1-WT D24		
Sample	R3	R4	R5	R7	R8	R9	R13	R14	R15	R16	R18	R20
Number of input reads	31032209	38721825	41062969	41329522	48173384	34654969	40663775	43288083	43777824	36839732	42077301	40412332
Average input read length	290	289	289	286	289	287	289	287	291	287	285	287
UNIQUE READS:												
Uniquely mapped reads number	28212739	35269250	37391651	36493625	42489124	31043675	37212612	39689179	39688522	32864419	36544527	35755699
Uniquely mapped reads %	**90.91 %**	**91.08 %**	**91.06 %**	**88.30 %**	**88.20 %**	**89.58 %**	**91.51 %**	**91.69 %**	**90.66 %**	**89.21 %**	**86.85 %**	**88.48 %**
Average mapped length	288.45	287.06	286.93	284.41	287.36	284.88	287.73	285.2	289.35	285.37	282.76	284.94

- Job Script:

```
#!/bin/bash
#SBATCH --job-name=mlh1STARmap.sh
#SBATCH -n 1
#SBATCH -N 1-1
#SBATCH -p all          #partition name
#SBATCH -c 12           #number of cores requested, must be
greater or equal to the number needed for the job
#SBATCH --mem=128G              #memory
#SBATCH -t 2-00         #days-hours:minutes runlimit after
which the job will be killed
#SBATCH -o %j.out               #std out file
#SBATCH -e %j.err               #std err file

module load Python/3.6.5
module load FastQC
module load STAR/2.7.9a

mkdir alignment_output

declare -a sample=("R3" "R4" "R5" "R7" "R8" "R9" "R13"
"R14" "R15" "R16" "R18" "R20")

for id in ${sample[@]}; do
        trim1=trimmedFQ/tr_${id}_R1_001.fastq.gz
        trim2=trimmedFQ/tr_${id}_R2_001.fastq.gz
        out=alignment_output/${id}_
        #echo "working on sample ${id}"

        #Map paired-end reads to reference after any pre-
processing steps (e.g., FASTQC).
        STAR --runThreadN 12 \
        --readFilesCommand zcat \
        --readFilesIn $trim1 $trim2 \
        --genomeDir /home/bosia/RNAseq/mm39_index \
        --outSAMtype BAM SortedByCoordinate \
        --outFileNamePrefix $out \
        --outSAMunmapped Within \
        --quantMode TranscriptomeSAM GeneCounts
done
```

Step 5: Optional/Alternative method, Count aligned reads using salmon

- First, generate a transcript file using gffread from the genome.fa and genome.gtf you indexed for the alignment with STAR:
  - gffread -w output.fa -g genome.fa genome.gtf

- Proceed with using Salmon to count
  - ***Version on HPC cluster is out of date. If you have errors, try downloading the newest version and running it locally rather than loading the HPC module
    - If you do this, remove the module load command from your job script and use ./salmon to call the local copy from the job submission directory.

- Job Script:

```
#!/bin/bash
#SBATCH --job-name=mlh1STARmap.sh
#SBATCH -n 1
#SBATCH -N 1-1
#SBATCH -p all          #partition name
#SBATCH -c 12           #number of cores requested, must be greater or equal to the
number needed for the job
#SBATCH --mem=128G              #memory
#SBATCH -t 2-00         #days-hours:minutes runlimit after which the job will be killed
#SBATCH -o %j.out               #std out file
#SBATCH -e %j.err               #std err file

module load Python/3.6.5
module load FastQC
module load STAR/2.7.9a
module load salmon/1.1.0
module load gffread

cd /home/bosia/RNAseq/GencodeM36
gffread -w mm39_transcripts.fa -g GRCm39.primary_assembly.genome.fa
gencode.vM36.primary_assembly.annotation.gtf
mv mm39_transcripts.fa ../mm39_index/
cd ..

mkdir salmon_output

declare -a sample=("R3" "R4" "R5" "R7" "R8" "R9" "R13" "R14" "R15" "R16" "R18" "R20")

for id in ${sample[@]}; do
        bam=alignment_output/${id}_Aligned.toTranscriptome.out.bam
        #echo "working on sample ${id}"

        #quantify STAR alignments using salmon
        salmon quant --threads 12 \
        --targets /home/bosia/RNAseq/mm39_index/mm39_transcripts.fa \
        --gencode \
        --libType ISR \
        --output salmon_output/$id \
        --alignments $bam
done
```

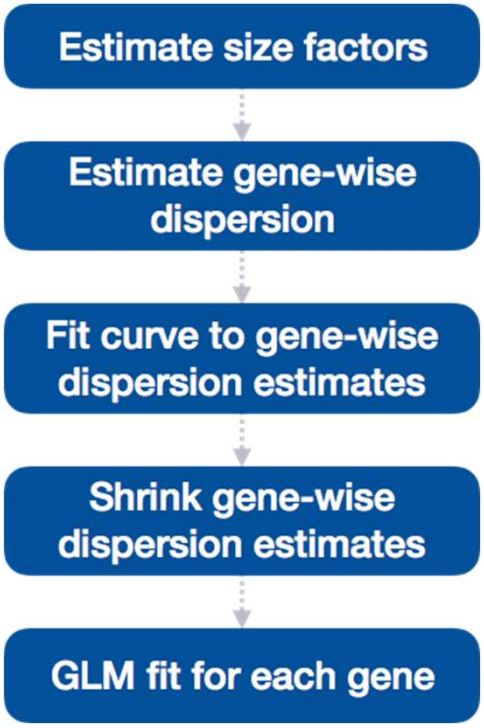
Step 6: Analyze read counts for differential expression using DeSeq2 (R package)

- First! Need to decide if you want to do analysis at the gene or transcript level.
  - If it's the gene level, output from STAR may be enough – Ended up going with this method!
  - If it's transcript, then use the Salmon output. Results for analysis by DeSeq2 will still be at gene level, but this method may give different results by taking transcripts into account – Did not end up trying this.

- Need to take read counts per gene/transcript from STAR/Salmon output and re-format for processing by R
  - Download data from cluster. Sequencing was unstranded PE reads, so take data from the left-most data column (Column $2 of 4).
  - Make csv files for each experimental condition for the count data:
    - D10 MLH1KO vs MLH1WT (R1-5 vs R11-15)
    - D24 MLH1KO vs MLH1WT (R6-10 vs R16-20)
    - MLH1WT D24 vs D10 (R16-20 vs R11-15)
    - MLH1KO D24 vs D10 (R6-10 vs R1-5)
  - Also make metadata files that provide information on experimental design.
  - Example:

id	dex	celltype	genotype
R3	treated	4T1	KO
R4	treated	4T1	KO
R5	treated	4T1	KO
R13	control	4T1	WT
R14	control	4T1	WT
R15	control	4T1	WT

DESeq2 pipeline:



(Count normalization step)

Dispersion = data variance

Dispersion fitting and shrinking is important to reduce false-positives

Can check dispersion plots to see if DESeq2 model fits the data well: plotDispEsts(dds)

Step 7: Perform additional post analysis on DESeq2 results.

- Try GSEA or GO analysis
  - Step 1: Need to make sure the correct mouse databases are being used in the R packages chosen for analysis.
  - Step 2: Use R package ClusterProfiler to perform GO enrichment analysis: EnrichGO
    - Minimum of 2 genes to be considered enriched (default is 5 genes).
  - Can use gene symbols or ENSEMBL IDs to perform analysis
    - Made tables using both, since results end up simplifying differently depending on which is used.
    - ENSEMBL IDs seem to give better simplified results (like/redundant pathways are grouped together).
    - Gene symbol results are still useful if you want to directly see what genes are involved in enriched GO terms.
    - Plots (following slides) are made from analysis performed using ENSEMBL IDs.