RNA-seq data cleaning

Paired-end fastq to counts

Kim Dill-McFarland, kadm@uw.edu

version April 27, 2022

Contents

Overview

This document covers the Hawn lab recommended data cleaning pipeline for RNA-seq fastq files. This contains the same steps as our SEAsnake pipeline, and we recommend you use that over running each steps individually here. However, if your data are not amenable to SEAsnake or you'd like more details on what goes on under the SEAsnake hood, please keep reading!

Specifically, this tutorial is for human, bulk, paired-end RNA-seq. Modification is needed for use with other organisms or single-read libraries; this pipeline cannot be applied to single cell data. This pipeline includes quality assessment and filtering, alignment, and count table generation. The detailed sections contain example code using a single example library. A full pipeline to process multiple libraries is available at the end.

0. Setup

Software

We process RNAseq libraries on a Linux server through AWS. This document is meant as an overview and bash code chunks herein should not be run in R. To achieve our AWS setup, please follow the AWS tutorials to setup an account and install conda software. Alternatively, you can install the following software individually on the system you are using.

- FastQC
- AdapterRemoval
- samtools
- bedtools
- Picard (optional)
- STAR
- Subread
- S3 fuse (if using AWS stored data)

Directory structure

Files will be organized as follows.

You can achieve this structure will the following.

```
mkdir -p data/
mkdir -p results/fastqc
mkdir -p results/fastq_trim
mkdir -p results/bam
mkdir -p results/metrics
mkdir -p results/counts
mkdir -p ref/

#set permissions
sudo chmod 777 -R results/
```

```
sudo chmod 777 -R ref/
```

The `data/` directory should contain all `.fastq`/`.fastq.gz` files for analysis and is likely fused to an S3 bucket. For reference, fusing can be completed with the following. See the AWS setup tutorial for more details.

```
# Key file for fuse. Fill in your access keys
echo ################:################ > ~/.passwd-s3fs
chmod 600  ~/.passwd-s3fs

# Fuse bucket to data/
s3fs bucket-name data -o passwd_file=~/.passwd-s3fs \
    -o default_acl=public-read -o uid=1000 -o gid=1000 -o umask=0007
```

Example library

This tutorial contains example outputs from one library in the Hawn MDM interferon stimulation RNAseq data set (Sept 2021). To run your file, please input the file and sample names here. Or use the script at the end to run all files in your `data/` directory.

If you have the ability to run on multiple threads, please also set this value to your total threads minus 2.

```
read1="6634-TH-1_S1_L005_R1_001.fastq.gz"
read2="6634-TH-1_S1_L005_R2_001.fastq.gz"
basename="6634-TH-1"

threads=1
```

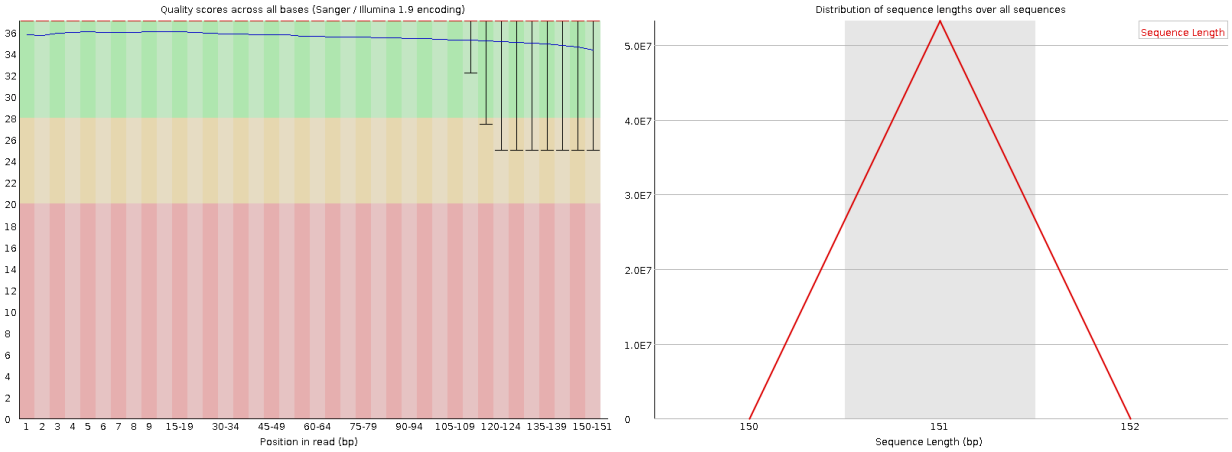
1. Sequence quality assessment

We assess sequence quality using FastQC for each read. Results are saved in `results/fastqc/` and include an `html` report with the following figures.

```
fastqc data/$read1 -o results/fastqc/ -t $threads

fastqc data/$read2 -o results/fastqc/ -t $threads
```

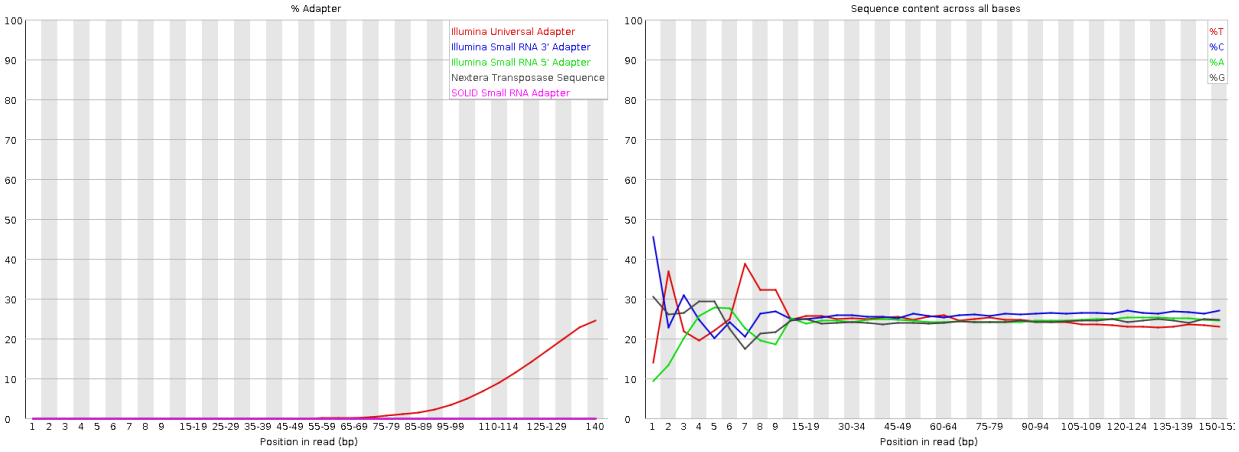
Example Overall, read 1 quality is high with Phred scores > 30 (left) and sequence lengths centered at the maximum for this run (right). The drop off in quality toward the 3' ends is common as sequencing error increases with sequence length.



There appear to be adapters present in these sequences. Here, FastQC identified the Illumina universal adapters toward the ends of sequences (left). This occurs when your biological sequence is shorter than the read length, and the sequencer goes beyond it into the adapter attaching DNA to the flow cell.

Not all adapters are identified by FastQC, so you can also check for them based on the sequence content (right). Adapters result in low diversity of per base content (*e.g.* percent of reads with each of ACTG at each position). This is seen here at the start of the sequences where the first 10 bp are disproportionately one or two base types.

Note that we do not see low diversity at the ends where we identified the universal adapters. This is because only up to ~25% of sequences have these adapters with the rest containing long enough amplicons to be entirely real data. Thus, per base content is not as impacted as it is with (likely) 100% of sequences having adapters at the start.



2. Adapter removal and quality filtering

Adapters are trimmed and sequences are quality filtered using AdapterRemoval. Trimmed files are saved in `results/fastq_trim`.

In these data, we have both 5' and 3' adapter contamination. Thus, quality filtering includes:

- `--trim5p` trim the 5' end by N base pairs to remove adapters. This value should be equal to where the per base sequence content levels out around 25% for all base types.
- `--adapter1 --adapter2` trim bp that align to known adapter sequences.
- `--maxns` remove sequences with > N ambiguous base

- `--trimqualities --minquality` trim ends until reach base with quality > N
- `--minlength` remove reads < N bp in length

```
AdapterRemoval --file1 data/$read1 --file2 data/$read2 \
    --basename results/fastq_trim/$basename --gzip \
    --trim5p 10 --maxns 1 --minlength 15 \
    --trimqualities --minquality 30 \
    --adapter1 AGATCGGAAGAGCACACGTCTGAACTCCAGTCAC \
    --adapter2 AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT \
    --threads $threads
```

Alternate adapter filtering

The above code will work for most Hawn data sets. However, you may need to make some modifications.

- If your barcodes are a different length, modify the number of bp trimmed with `--trim5p`.
- If you do not have barcode contamination, remove the `--trim5p` option.
- If you do not have adapter contamination, the original code will work but you can improve speed by removing the `--adapter1 --adapter2` options.
- If you have different adapters, modify the sequences after `--adapter1 --adapter2`. You can determine adapter sequences for several common library preparations here or by using

  ```
  AdapterRemoval --identify-adapters --file1 $read1 --file2 $read2 --threads $threads
  ```

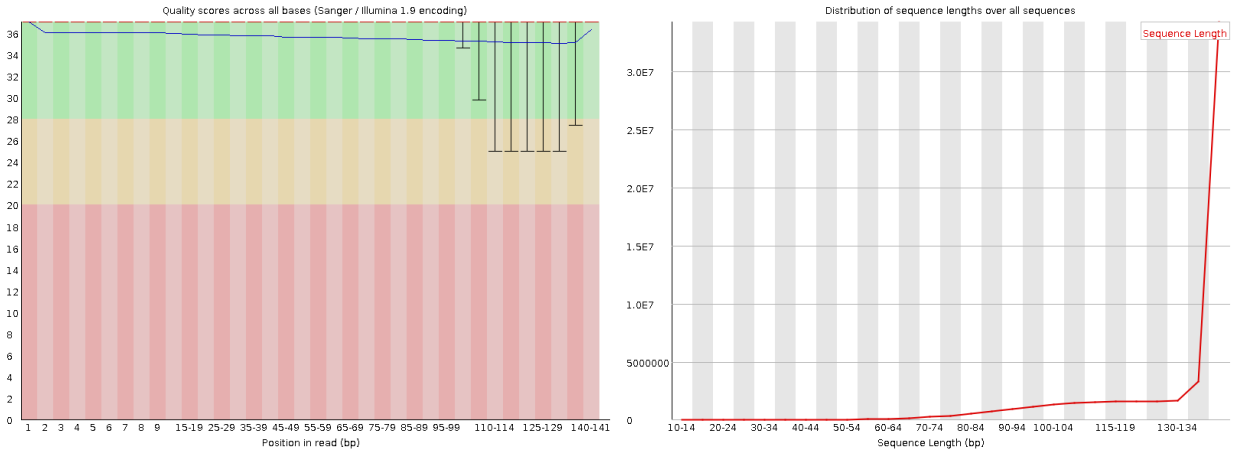
Re-assess quality

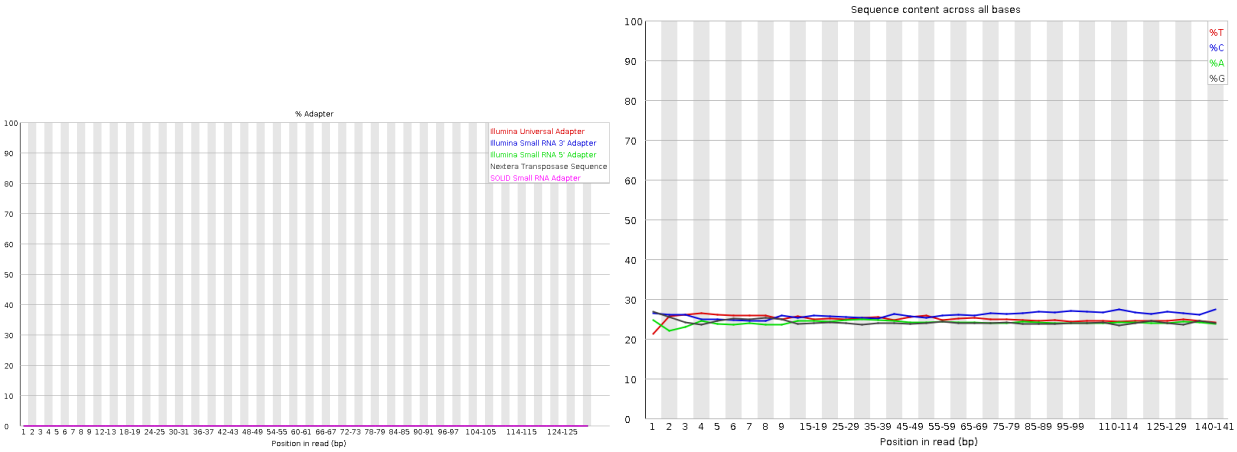
Next, we assess trimmed sequence quality using FastQC again. Results are saved in `results/fastqc/`.

```
fastqc results/fastq_trim/$basename.pair1.truncated.gz -o results/fastqc/ -t $threads

fastqc results/fastq_trim/$basename.pair2.truncated.gz -o results/fastqc/ -t $threads
```

Example Read 1 sequence quality is improved and adapters are no longer present in the data.





3. Alignment to reference genome

Using STAR, we align sequence data to the GRCh38 reference human genome. Results are saved in `results/bam/`.

Format reference genome

Look for the latest genome release at https://ftp.ensembl.org/pub/ and set the release number here.

```
number="104"
```

Then, download the reference genome and create an alignment index.

```
#Set file structure
mkdir -p ref/release"$number"/STARref
sudo chmod 777 -R ref/

#Download reference
source_url="ftp://ftp.ensembl.org/pub/release-$number"

sudo curl -O --output-dir ref/release"$number"/STARref \
    $source_url/gtf/homo_sapiens/Homo_sapiens.GRCh38.$number.gtf.gz

sudo curl -O --output-dir ref/release"$number"/STARref \
    $source_url/fasta/homo_sapiens/dna/Homo_sapiens.GRCh38.dna.primary_assembly.fa.gz

gunzip ref/release"$number"/STARref/*

#Index genome
STAR --runMode genomeGenerate \
     --genomeDir ref/release$number/STARindex \
     --genomeFastaFiles \
        ref/release$number/STARref/Homo_sapiens.GRCh38.dna.primary_assembly.fa \
     --sjdbGTFfile ref/release$number/STARref/Homo_sapiens.GRCh38.$number.gtf \
     --sjdbOverhang 99 \
     --runThreadN $threads
```

Hawn lab members: Check the S3 bucket `human-ref` for the latest release. If it is present, simply fuse this bucket to your `ref/` directory.

```
s3fs human-ref ref -o passwd_file=~/.passwd-s3fs \
    -o default_acl=public-read -o uid=1000 -o gid=1000 -o umask=0007
```

Align sequences

Now we align sequences to the reference index and save the alignment in a compressed BAM file sorted by genome position.

```
STAR --genomeDir ref/release$number/STARindex \
         --readFilesIn results/fastq_trim/$basename.pair1.truncated.gz \
                       results/fastq_trim/$basename.pair2.truncated.gz \
         --readFilesCommand zcat \
         --outFileNamePrefix results/bam/"$basename"_ \
         --outSAMtype BAM SortedByCoordinate \
         --runThreadN $threads \
         --runRNGseed 8756
```

4. Alignment quality assessment and filtering

Alignments are filtered with samtools `view` with the following flags. Results are saved in `results/bam/`. If you require different filtering, Broad has a nice tool for determining flag values.

- `-h` keep header
- `-f 3` keep paired reads where both mapped
- `-F 1284` remove unmapped reads, non-primary alignments, and PCR duplicates
- `-q 30` remove alignments with MAPQ < 30

```
samtools view results/bam/"$basename"_Aligned.sortedByCoord.out.bam \
      -h -f 3 -F 1284 -q 30 -@ $threads \
      -o results/bam/"$basename"_filter.bam
```

Then, we assess alignment quality before and after filtering with samtools `flagstat`. Results are saved in `results/metrics/`.

```
samtools flagstat results/bam/"$basename"_Aligned.sortedByCoord.out.bam \
    -@ $threads > results/metrics/"$basename"_Aligned_flagstat.tsv

samtools flagstat results/bam/"$basename"_filter.bam \
    -@ $threads > results/metrics/"$basename"_filter_flagstat.tsv
```

Example `flagstat` results are formatted as total reads passing QC + total reads failing QC. Since we filtered separately with `view` and did not input any QC into `flagstat`, all our failed values should be 0, and we need only look at the first values.

Here, we see that 5.453667×10^6 sequences were removed during alignment filtering (1). These were mainly secondary alignments (2), unpaired reads (3 compared to totals in 1), and PCR duplicates (4). After filtering, read 1 contains about 56 million reads

Ref	Aligned	Aligned, filtered	Flag
1	61871001 + 0	56417334 + 0	in total (QC-passed reads + QC-failed reads)
2	3052746 + 0	0 + 0	secondary

Ref	Aligned	Aligned, filtered	Flag
.	0 + 0	0 + 0	supplementary
.	0 + 0	0 + 0	duplicates
.	61871001 + 0	56417334 + 0	mapped (100.00% : N/A)
.	58818255 + 0	56417334 + 0	paired in sequencing
.	29409871 + 0	28208667 + 0	read1
.	29408384 + 0	28208667 + 0	read2
3	58816556 + 0	56417334 + 0	properly paired (100.00% : N/A)
3	58816556 + 0	56417334 + 0	with itself and mate mapped
4	1699 + 0	0 + 0	singletons (0.00% : N/A)
.	0 + 0	0 + 0	with mate mapped to a different chr
.	0 + 0	0 + 0	with mate mapped to a different chr (mapQ>=5)

Optional alignment assessment

Picard is another tool useful for assessing alignment quality. Picard metrics include median coefficient of variation (CV) of coverage, alignment percentage, etc. While these metrics are useful, this program is very slow and cannot run on multiple threads. Thus, you may wish to skip this assessment for large data sets.

Download Picard reference. **Hawn lab members**, this reference is already in your `ref/` directory if you previously fused to the `human-ref` bucket.

```
#Set file structure
mkdir -p ref/PICARDref
sudo chmod 777 -R ref/

#Download reference
sudo curl -O --output-dir ref/PICARDref \
    http://hgdownload.cse.ucsc.edu/goldenPath/hg38/database/refFlat.txt.gz

gunzip ref/PICARDref/refFlat.txt.gz

## Remove "chr" in chromosome name to match ensembl alignment using in STAR
sed 's/chr//' ref/PICARDref/refFlat.txt > ref/PICARDref/refFlat.ensembl.txt
```

Then, run Picard `CollectRnaSeqMetrics`.

Note: Your Picard executable will have a different file path if you did not install with conda. Also, please update to the latest Picard version as necessary.

```
java -jar ~/apps/anaconda/share/picard-2.26.2-0/picard.jar \
        CollectRnaSeqMetrics \
        REF_FLAT=ref/PICARDref/refFlat.ensembl.txt \
        INPUT=results/bam/"$basename"_Aligned.sortedByCoord.out.bam  \
        OUTPUT=results/metrics/"$basename"_Aligned_picard.tsv \
        ASSUME_SORTED=true STRAND_SPECIFICITY=NONE MINIMUM_LENGTH=500 \
        QUIET=true VERBOSITY=ERROR
```

Example Below are all the metrics Picard calculates. We generally only use `MEDIAN_CV_COVERAGE`, which is a measure of coverage with 0 being ideal (no variation in coverage across all genes in the genome) and greater than 1 being poor (lots of variation in coverage).

Metric	Value
PF_BASES	8208538047
PF_ALIGNED_BASES	8133215003
RIBOSOMAL_BASES	NA
CODING_BASES	3544622408
UTR_BASES	2415868363
INTRONIC_BASES	1890267853
INTERGENIC_BASES	282456379
IGNORED_READS	0.000000
CORRECT_STRAND_READS	0.000000
INCORRECT_STRAND_READS	0.000000
NUM_R1_TRANSCRIPT_STRAND_READS	154837
NUM_R2_TRANSCRIPT_STRAND_READS	17949451
NUM_UNEXPLAINED_READS	994151
PCT_R1_TRANSCRIPT_STRAND_READS	0.008553
PCT_R2_TRANSCRIPT_STRAND_READS	0.991447
PCT_RIBOSOMAL_BASES	NA
PCT_CODING_BASES	0.435821
PCT_UTR_BASES	0.297037
PCT_INTRONIC_BASES	0.232413
PCT_INTERGENIC_BASES	0.034729
PCT_MRNA_BASES	0.732858
PCT_USABLE_BASES	0.726133
PCT_CORRECT_STRAND_READS	0.000000
MEDIAN_CV_COVERAGE	0.658344
MEDIAN_5PRIME_BIAS	0.332954
MEDIAN_3PRIME_BIAS	0.159626
MEDIAN_5PRIME_TO_3PRIME_BIAS	1.556895
SAMPLE	NA
LIBRARY	NA
READ_GROUP	NA

5. Count table generation

Finally, we count reads in gene exons with Subread `featureCounts`. This function can also explore any sequence type listed in the `gtf` file, though exons are the standard type for differential expression analysis.

```
featureCounts -T $threads -g gene_id -t exon -p \
  -a ref/release$number/STARref/Homo_sapiens.GRCh38.$number.gtf \
  -o results/counts/MDM-IFN.featurecounts.tsv \
  results/bam/*filter.bam
```

This counts table is used for subsequent analyses such as differential gene expression, gene module building, gene set enrichment, etc. as seen in the next tutorials.

Batch script

This bash script will run the above pipeline for all `.fastq` files in a directory. You should complete the setup variables at the beginning specific to your data.

```
#!/bin/bash

##### Bulk paired-end RNAseq data cleaning ######
```

```
########################################
## Setup
########################################
# Directory names must NOT end in /

project_name="MDM-IFN"
data_dir="data"
threads=30
adapter_length=10
number=104

# Create directory structure
mkdir -p results/fastqc
mkdir -p results/fastq_trim
mkdir -p results/bam
mkdir -p results/metrics
mkdir -p results/counts
mkdir -p ref/

#set permissions
sudo chmod 777 -R results/
sudo chmod 777 -R ref/

########################################
## Quality assessment 1
########################################

for file in $data_dir/*fastq.gz ;
do
  fastqc $file -o results/fastqc/ -t $threads
done

########################################
## Adapter removal
########################################

paste <(ls $data_dir/*R1_001.fastq.gz) \
      <(ls $data_dir/*R2_001.fastq.gz) |

while read file1 file2;
do
  name=$(paste -d '\0' \
            <(echo 'results/fastq_trim/') \
            <(awk -F'[_]S' '{print $1}' <(basename $file1)))

  AdapterRemoval --file1 $file1 --file2 $file2 \
    --basename $name --gzip \
    --trim5p $adapter_length --maxns 1 --minlength 15 \
    --trimqualities --minquality 30 \
    --adapter1 AGATCGGAAGAGCACACGTCTGAACTCCAGTCAC \
    --adapter2 AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT \
    --threads $threads
done
```

```
########################################
## Quality assessment 2
########################################

for file in results/fastq_trim/*pair[12].truncated.gz ;
do
  fastqc $file -o results/fastqc/ -t $threads
done

########################################
## Alignment reference
########################################

#Set file structure
mkdir -p ref/release"$number"/STARref
sudo chmod 777 -R ref/

#Download reference
sudo curl -O --output-dir ref/release"$number"/STARref \
    ftp://ftp.ensembl.org/pub/release-$number/gtf/homo_sapiens/Homo_sapiens.GRCh38.$number.gtf.gz

sudo curl -O --output-dir ref/release"$number"/STARref \
    ftp://ftp.ensembl.org/pub/release-$number/fasta/homo_sapiens/dna/Homo_sapiens.GRCh38.dna.primary_ass

gunzip ref/release"$number"/STARref/*

#Index genome
STAR --runMode genomeGenerate \
     --genomeDir ref/release$number/STARindex \
     --genomeFastaFiles \
        ref/release$number/STARref/Homo_sapiens.GRCh38.dna.primary_assembly.fa \
     --sjdbGTFfile ref/release$number/STARref/Homo_sapiens.GRCh38.$number.gtf \
     --sjdbOverhang 99 \
     --runThreadN $threads

#Move log file to within reference directory
mv Log.out ref/release$number/STARindex/

########################################
## Alignment
########################################
paste <(ls results/fastq_trim/*.pair1.truncated.gz) \
      <(ls results/fastq_trim/*.pair2.truncated.gz) |

while read file1 file2;
do
    echo "Aligning" $(basename  -- "$file1");

    name=$(paste -d '\0' \
            <(echo 'results/bam/') \
            <(awk -F'[.]pair' '{print $1}' <(basename $file1)) \
            <(echo '_'))
```

```
    STAR --genomeDir ref/release$number/STARindex \
         --readFilesIn $file1 $file2 \
         --readFilesCommand zcat \
         --outFileNamePrefix $name \
         --outSAMtype BAM SortedByCoordinate \
         --runThreadN $threads \
         --runRNGseed 8756
done

########################################
## Quality filter alignment
########################################

for file in results/bam/*_Aligned.sortedByCoord.out.bam ;
do
  name=$(paste -d '\0' \
            <(echo 'results/bam/') \
            <(awk -F'[_]Aligned' '{print $1}' <(basename $file)) \
            <(echo '_filter.bam'))

  samtools view $file -h -f 3 -F 1284 -q 30 -@ $threads > $name
done

########################################
## flagstat
########################################
for file in results/bam/*.bam ;
do
  name=$(paste -d '\0' \
            <(echo 'results/metrics/') \
            <(awk -F'[.]' '{print $1}' <(basename $file)) \
            <(echo '_flagstat.tsv'))

  samtools flagstat -@ $threads $file > $name
done

########################################
## Picard
########################################
#Set file structure
mkdir -p ref/PICARDref
sudo chmod 777 -R ref/

#Download reference
sudo curl -O --output-dir ref/PICARDref \
    http://hgdownload.cse.ucsc.edu/goldenPath/hg38/database/refFlat.txt.gz

gunzip ref/PICARDref/refFlat.txt.gz

## Remove "chr" in chromosome name to match ensembl alignment using in STAR
sed 's/chr//' ref/PICARDref/refFlat.txt > ref/PICARDref/refFlat.ensembl.txt

for file in results/bam/*sortedByCoord.out.bam ;
```

```
do
  name=$(paste -d '\0' \
            <(echo 'results/metrics/') \
            <(awk -F'[.]' '{print $1}' <(basename $file)) \
            <(echo '_picard.tsv'))


    java -jar ~/apps/anaconda/share/picard-2.26.2-0/picard.jar \
        CollectRnaSeqMetrics \
        REF_FLAT=ref/PICARDref/refFlat.ensembl.txt \
        INPUT=$file OUTPUT=$name \
        ASSUME_SORTED=true STRAND_SPECIFICITY=NONE MINIMUM_LENGTH=500 \
        QUIET=true VERBOSITY=ERROR
done

########################################
## Count reads in genes
########################################

featureCounts -T $threads -g gene_id -t exon -p \
  -a ref/release$number/STARref/Homo_sapiens.GRCh38.$number.gtf \
  -o results/counts/$project_name.featurecounts.tsv \
  results/bam/*filter.bam

################## END ##################
```