# Introduction to Bulk RNAseq data analysis

Quality Control

## Contents



## Checking the quality of the data

Once we have aligned our reads and quantified gene expression with Salmon, it is then possible to run some additional quality checks on our data. We will use a tool call Picard to gather some information about duplication rates and the distribution of reads within each gene in order to assess RNA integrity. We will also gather some QC metrics from the Salmon outputs, such as alignment rate and insert size.

### Detailed metrics with Picard Tools

**Picard Tools** is a suite of tools for analysing and manipulating sequencing data. It is maintained by the Broad Institute and comprises many different tools for doing jobs such as generating QC metrics, modifying bam files in various ways, or converting files between different formats.

Picard is a java based programme and so to run a particular Picard tool the general format for the command is:

```
java -jar picard/picard.jar PicardToolName OPTION1=value1 OPTION2=value2 ...
```

- `picard.jar` is the **J**ava **AR**chive file that contains all the tools. It can be found in the *picard* directory under *Course_Materials*.
- `PicardToolName` is the name of the tools that you wish to use.

This is then followed by a series of options/arguments for that particular tool. Each tool has specific options and arguments and you can check these most easily by going to the Picard website, or by using the `--help` command:

```
java -jar picard/picard.jar PicardToolName --help
```

#### Duplication metrics

The first tool we are going to use is `MarkDuplicates`. This tool actually performs two tasks.

First, Picard reads through the bam file and finds any duplicate reads - these are reads with the same genomic location at the 5' end. For each group of duplicate reads, Picard selects a "primary" read based on the base quality scores and then marks all other reads in the group as "duplicates". For paired end reads, the entire fragment is considered, this essentially means both ends must be duplicated for the read pair to be marked as a duplicate.

Once the duplicate reads are marked, Picard outputs a new bam file in which the duplicate reads are marked and then also generates a metrics file that contains information about the duplication rate.

**Exercise 1**

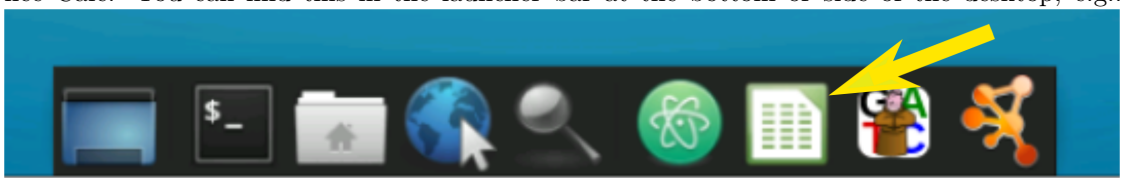
For the purposes of this exercise we will work with a bam file that only contains 2 million reads. Running Picard on the full bam file would take too long for the practical. You should find a file called `SRR7657883.salmon.sorted.bam` under the directory `salmon_qc_demo/SRR7657883`, please use this file rather than the bam file you created in the previous session.

1. Run Picard's MarkDuplicates tool on the sorted bam file using the following command:

```
java -jar picard/picard.jar MarkDuplicates \
          INPUT= salmon_qc_demo/SRR7657883/SRR7657883.salmon.sorted.bam \
          OUTPUT=salmon_qc_demo/SRR7657883/SRR7657883.salmon.mkdup.bam \
          METRICS_FILE=salmon_qc_demo/SRR7657883/SRR7657883.mkdup_metrics.txt \
          CREATE_INDEX=true \
          VALIDATION_STRINGENCY=SILENT
```

**Note**: The \ at the end of each line tells the terminal that when you press `Enter`, you have not yet finished typing the command. You can if you wish, type the whole command on a single line, omitting the \. The command is written across multiple lines here just to make it easier to read.

Q. What is the duplication rate for this bam file? You'll need to look at the metrics file. The easiest way is to open it in a spreadsheet. On the course machines we have LibreOffice Calc. You can find this in the launcher bar at the bottom or side of the desktop, e.g.:



You can find details about the contents of the metrics file in the Picard documentation.

**RNA alignment metrics**

The `CollectRnaSeqMetrics` tool produces metrics describing the distribution of the reads across different genomic locations - intronic, exonic, intergenic, UTR - and the distribution of bases within the transcripts. In our case the reads have been aligned to the transcriptome, so we can not determine the genomic location, however, the normalised coverage across transcripts is useful information for assessing the integrity of the original RNA.

The `CollectRnaSeqMetrics` requires four pieces of information to run:

1. The input bam file
2. A file name for the output metrics file
3. A file containing gene annotations in a format called **RefFlat**, which is defined here.
4. A parameter for strand specificity. Strand specificity is determined by the library prep chemistry. There are three possible options:
   a) FIRST_READ_TRANSCRIPTION_STRAND - Forward stranded library prep, i.e. the reads are on the transcription strand
   b) SECOND_READ_TRANSCRIPTION_STRAND - Reverse stranded library prep, i.e. the reads are on the reverse strand
   c) NONE - Unstranded library prep, i.e. the reads may be on either strand
   With your own data you would need to find this information out from whoever has prepared the library. In this case our library prep is unstranded, so we will use `NONE`.

**Generate the RefFlat file** The first thing we will need to do is construct the RefFlat file. If we had aligned our reads to the genome, each read in our bam file would have a genomic location - a chromosome and a position on that chromosome. We would then need an annotation file that contained the genomic

locations of the genes in the genome in order to translate the genomic locations of the reads into gene based information.

The RefFlat file is one of a variety of formats for describing gene annotation. In RefFlat format each line contains the information for a single transcript, including contig/sequence (chromosome) on which the gene is located, the start and end locations on the conig for the transcript, and the start and end locations for each exon of the trancscript. The Picard tool `CollectRnaSeqMetrics` normally uses this refFlat file to translate the genomic alignments of reads into details of the gene encoded by the transcript molecule from which the reads originated.

To analyse the alignments from Salmon, we still need to provide a refFlat file, but in this case each contig/sequence in the reference is a transcript with a single "exon" (the transcriptome fasta does not include introns). We can use the sequence information that is stored in the header of the bam file to generate this RefFlat file.

We can do this using the data contained in the bam file header. The file `create_refflat_from_sam_header.py` in the **scripts** directory is a python script that will read a Salmon mappings BAM file and generate the required RefFlat file from the header.

For future reference, you can download the script from our GitHub repository.In addition to the python script, you will also need to have Python and samtools installed and on the PATH.

`create_refflat_from_sam_header.py` requires two parameters:

- `-b` - the bam file to use
- `-o` - the output file name

```
./scripts/create_refflat_from_sam_header.py \
  -b salmon_qc_demo/SRR7657883/SRR7657883.salmon.sorted.bam \
  -o references/GRCm38_transriptome_refFlat.txt
```

Note that we only have to do this once, we do not need to create a separate refFlat file for each sample as they have all been quantified using the same transcriptome reference.

**Exercise 2**

1. Run Picard's `CollectRnaSeqMetrics` tool on the sorted bam file using the following arguments:
   - INPUT - The bam file
   - OUTPUT - `salmon_qc_demo/SRR7657883/SRR7657883.salmon.RNA_metrics.txt`
   - REF_FLAT - the refFlat reference file
   - STRAND - `NONE`
   - VALIDATION_STRINGENCY - SILENT

Answer

```
java -jar picard/picard.jar CollectRnaSeqMetrics \
          INPUT=salmon_qc_demo/SRR7657883/SRR7657883.salmon.sorted.bam \
          OUTPUT=salmon_qc_demo/SRR7657883/SRR7657883.salmon.RNA_metrics.txt \
          REF_FLAT=references/GRCm38_transriptome_refFlat.txt \
          STRAND=NONE \
          VALIDATION_STRINGENCY=SILENT
```

**Note**: The `\` at the end of each line tells the terminal that when you press `Enter`, you have not yet finished typing the command. You can if you wish, type the whole command on a single line, omitting the `\`. The command is written across multiple lines here just to make it easier to read.

The results of this analysis are best viewed graphically, we will do this in the next exercise.

## Visualising QC results with MultiQC

MultiQC is a tool for collating multiple QC results files into a single report. Its use is simple, you just run the command `multiqc` on the directory containing your metrics files. The general command is:

```
multiqc <directory containing metrics files>
```

We will add a couple of options to control the output directory and the name of the report.

### Exercise 3

1. Run multiqc on the salmon directory:

```
multiqc \
    -n Salmon_QC_Report.html \
    -o salmon_qc_demo \
    salmon_qc_demo
```

- `-n` - a name for the report
- `-o` - the directory in which to place the report

2. Open the html report that was generated by multiqc and inspect the QC plots The easiest way to do this is type `xdg-open salmon_qc_demo/Salmon_QC_Report.html`, which will open the report in a web browser.

### Exercise 4

In the `salmon` directory you should find Salmon outputs, duplication metrics and RNAseq metrics for all of the samples from the study.

1. Run multiqc on the contents of the whole `salmon` directory.

2. Open the html report that was generated by multiqc and inspect the QC plots

Q. Are there any samples that look problematic?

---