

# nf-core/rnaseq: Output

## Introduction

This document describes the output produced by the pipeline. Most of the plots are taken from the MultiQC report generated from the [full-sized test dataset](#) for the pipeline using a command similar to the one below:

```
nextflow run nf-core/rnaseq -profile test_full,<docker/singularity>
```

The directories listed below will be created in the results directory after the pipeline has finished. All paths are relative to the top-level results directory.

## Pipeline overview

The pipeline is built using [Nextflow](#) and processes data using the following steps:

- [Preprocessing](#)
  - [cat](#) - Merge re-sequenced FastQ files
  - [FastQC](#) - Raw read QC
  - [UMI-tools extract](#) - UMI barcode extraction
  - [TrimGalore](#) - Adapter and quality trimming
  - [BBSplit](#) - Removal of genome contaminants
  - [SortMeRNA](#) - Removal of ribosomal RNA
- [Alignment and quantification](#)
  - [STAR and Salmon](#) - Fast spliced aware genome alignment and transcriptome quantification
  - [STAR via RSEM](#) - Alignment and quantification of expression levels
  - [HISAT2](#) - Memory efficient splice aware alignment to a reference
- [Alignment post-processing](#)
  - [SAMtools](#) - Sort and index alignments
  - [UMI-tools dedup](#) - UMI-based deduplication
  - [picard MarkDuplicates](#) - Duplicate read marking
- [Other steps](#)
  - [StringTie](#) - Transcript assembly and quantification
  - [BEDTools and bedGraphToBigWig](#) - Create bigWig coverage files
- [Quality control](#)
  - [RSeQC](#) - Various RNA-seq QC metrics
  - [Qualimap](#) - Various RNA-seq QC metrics
  - [dupRadar](#) - Assessment of technical / biological read duplication
  - [Preseq](#) - Estimation of library complexity
  - [featureCounts](#) - Read counting relative to gene biotype
  - [DESeq2](#) - PCA plot and sample pairwise distance heatmap and dendrogram
  - [MultiQC](#) - Present QC for raw reads, alignment, read counting and sample similarity
- [Pseudo-alignment and quantification](#)
  - [Salmon](#) - Wicked fast gene and isoform quantification relative to the transcriptome
- [Workflow reporting and genomes](#)
  - [Reference genome files](#) - Saving reference genome indices/files
  - [Pipeline information](#) - Report metrics generated during the workflow execution

## Preprocessing

### cat

#### ► Output files

If multiple libraries/runs have been provided for the same sample in the input samplesheet (e.g. to increase sequencing depth) then these will be merged at the very beginning of the pipeline in order to have consistent sample naming

## Table of Contents

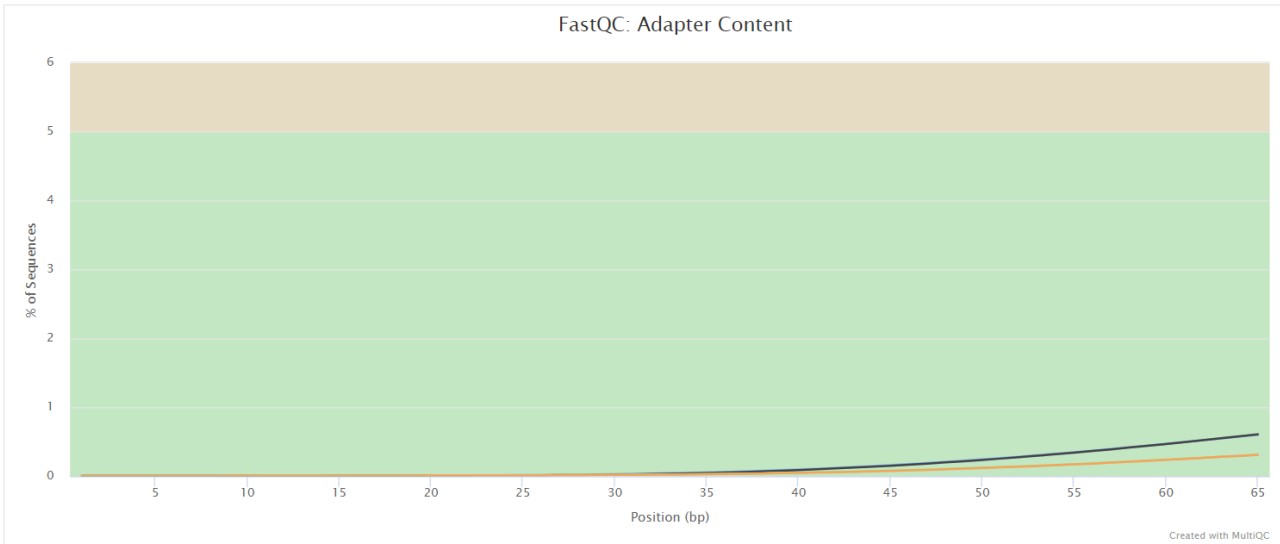
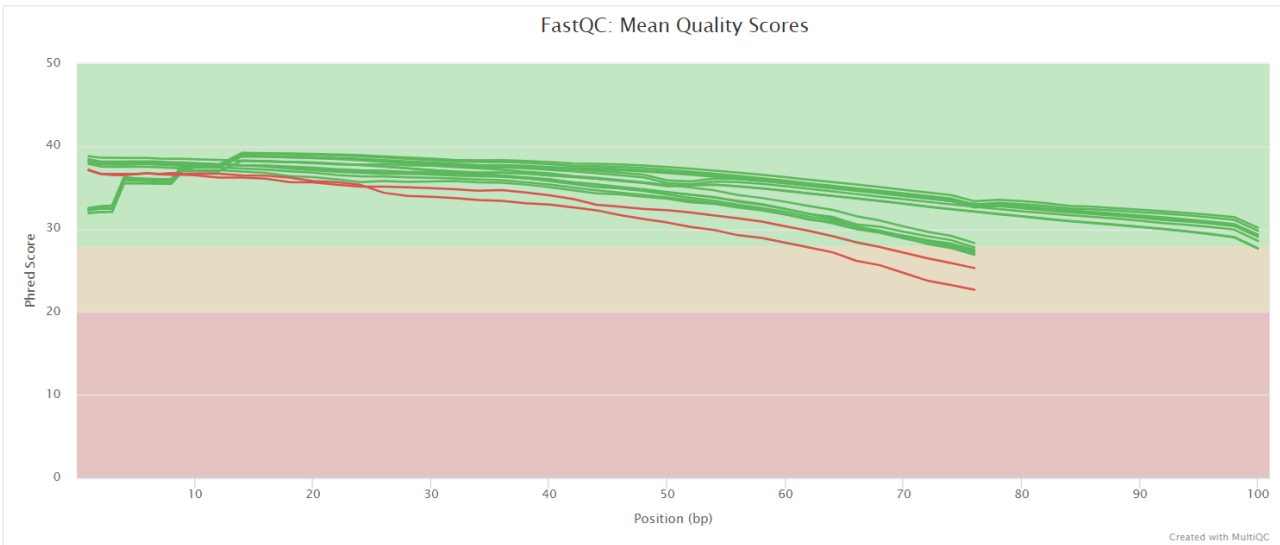
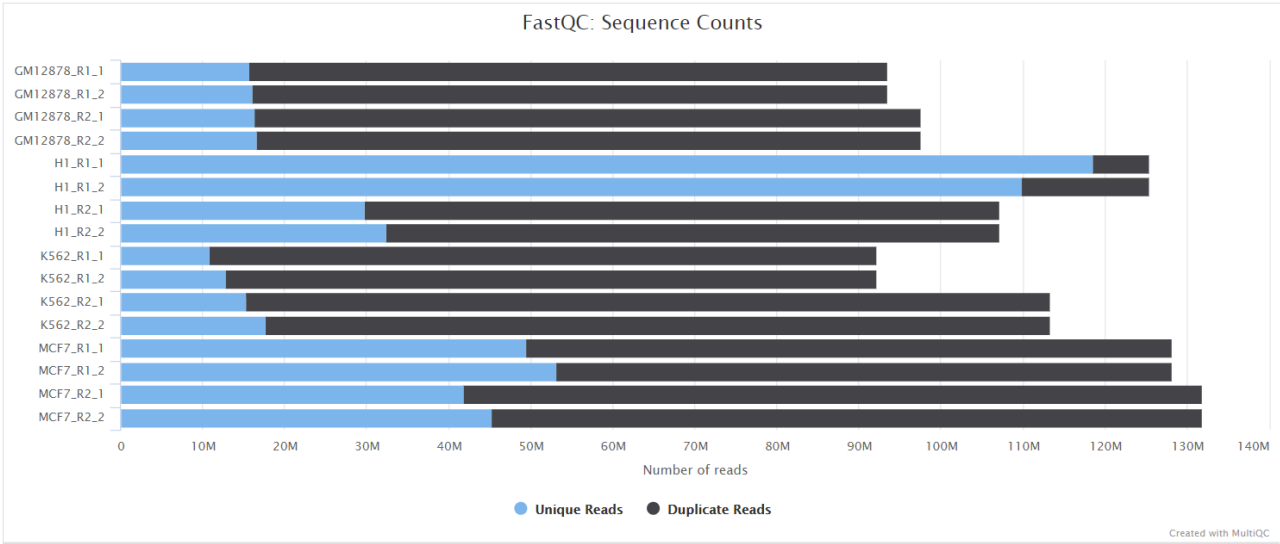
- [Introduction](#)
- [Pipeline overview](#)
- [Preprocessing](#)
  - [cat](#)
  - [FastQC](#)
  - [UMI-tools extract](#)
  - [TrimGalore](#)
  - [BBSplit](#)
  - [SortMeRNA](#)
- [Alignment and quantification](#)
  - [STAR and Salmon](#)
  - [STAR via RSEM](#)
  - [HISAT2](#)
- [Alignment post-processing](#)
  - [SAMtools](#)
  - [UMI-tools dedup](#)
  - [picard MarkDuplicates](#)
- [Other steps](#)
  - [StringTie](#)
  - [BEDTools and bedGraphToBigWig](#)
- [Quality control](#)
  - [RSeQC](#)
    - [Infer experiment](#)
    - [Read distribution](#)
    - [Junction annotation](#)
    - [Inner distance](#)
    - [Junction saturation](#)
    - [Read duplication](#)
    - [BAM stat](#)
    - [TIN](#)
  - [Qualimap](#)
  - [dupRadar](#)
  - [Preseq](#)
  - [featureCounts](#)
  - [DESeq2](#)
  - [MultiQC](#)
- [Pseudo-alignment and quantification](#)
  - [Salmon](#)
- [Workflow reporting and genomes](#)
  - [Reference genome files](#)
  - [Pipeline information](#)

throughout the pipeline. Please refer to the [usage documentation](#) to see how to specify these samples in the input samplesheet.

## FastQC

► Output files

[FastQC](#) gives general quality metrics about your sequenced reads. It provides information about the quality score distribution across your reads, per base sequence content (%A/T/G/C), adapter contamination and overrepresented sequences. For further reading and documentation see the [FastQC help pages](#).



## UMI-tools extract

### ► Output files

[UMI-tools](#) deduplicates reads based on unique molecular identifiers (UMIs) to address PCR-bias. Firstly, the UMI-tools `extract` command removes the UMI barcode information from the read sequence and adds it to the read name. Secondly, reads are deduplicated based on UMI identifier after mapping as highlighted in the [UMI-tools dedup](#) section.

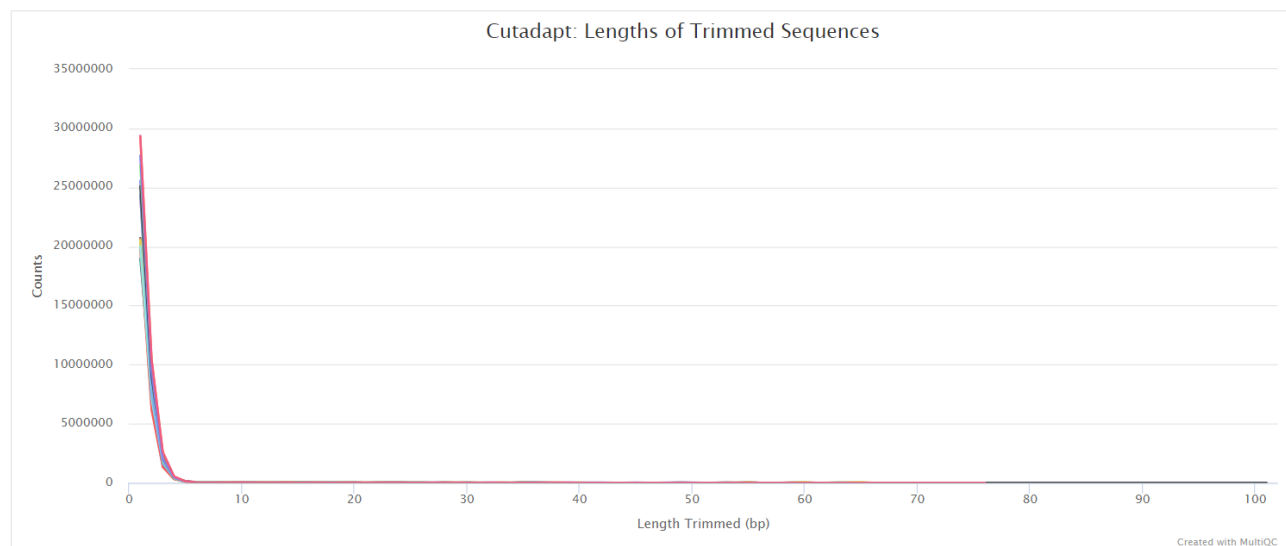
To facilitate processing of input data which has the UMI barcode already embedded in the read name from the start, `--skip_umi_extract` can be specified in conjunction with `--with_umi`.

## TrimGalore

### ► Output files

[Trim Galore!](#) is a wrapper tool around Cutadapt and FastQC to perform quality and adapter trimming on FastQ files. By default, Trim Galore! will automatically detect and trim the appropriate adapter sequence.

**NB:** TrimGalore! will only run using multiple cores if you are able to use more than > 5 and > 6 CPUs for single- and paired-end data, respectively. The total cores available to TrimGalore! will also be capped at 4 (7 and 8 CPUs in total for single- and paired-end data, respectively) because there is no longer a run-time benefit. See [release notes](#) and [discussion whilst adding this logic to the nf-core/atacseq pipeline](#).



## BBSplit

### ► Output files

[BBSplit](#) is a tool that bins reads by mapping to multiple references simultaneously, using BBMap. The reads go to the bin of the reference they map to best. There are also disambiguation options, such that reads that map to multiple references can be binned with all of them, none of them, one of them, or put in a special "ambiguous" file for each of them.

This functionality would be especially useful, for example, if you have [mouse PDX](#) samples that contain a mixture of human and mouse genomic DNA/RNA and you would like to filter out any mouse derived reads.

The BBSplit index will have to be built at least once with this pipeline by providing `--bbsplit_fasta_list` which has to be a file containing 2 columns: short name and full path to reference genome(s):

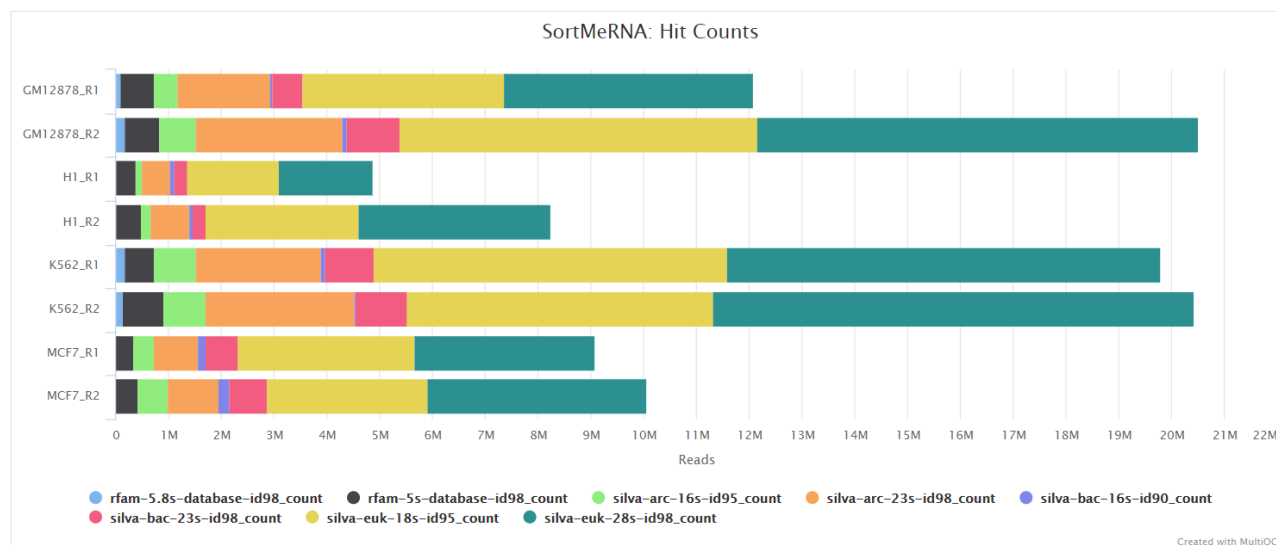
```
mm10,/path/to/mm10.fa
ecoli,/path/to/ecoli.fa
sarscov2,/path/to/sarscov2.fa
```

You can save the index by using the `--save_reference` parameter and then provide it via `--bbsplit_index` for future runs. As described in the [Output files](#) dropdown box above the FastQ files relative to the main reference genome will always be called `*primary*.fastq.gz`.

## SortMeRNA

### ► Output files

When `--remove_ribo_rna` is specified, the pipeline uses [SortMeRNA](#) for the removal of ribosomal RNA. By default, [rRNA databases](#) defined in the SortMeRNA GitHub repo are used. You can see an example in the pipeline Github repository in `assets/rRNA-default-dbs.txt` which is used by default via the `--ribo_database_manifest` parameter. Please note that commercial/non-academic entities require [licensing for SILVA](#) for these default databases.



## Alignment and quantification

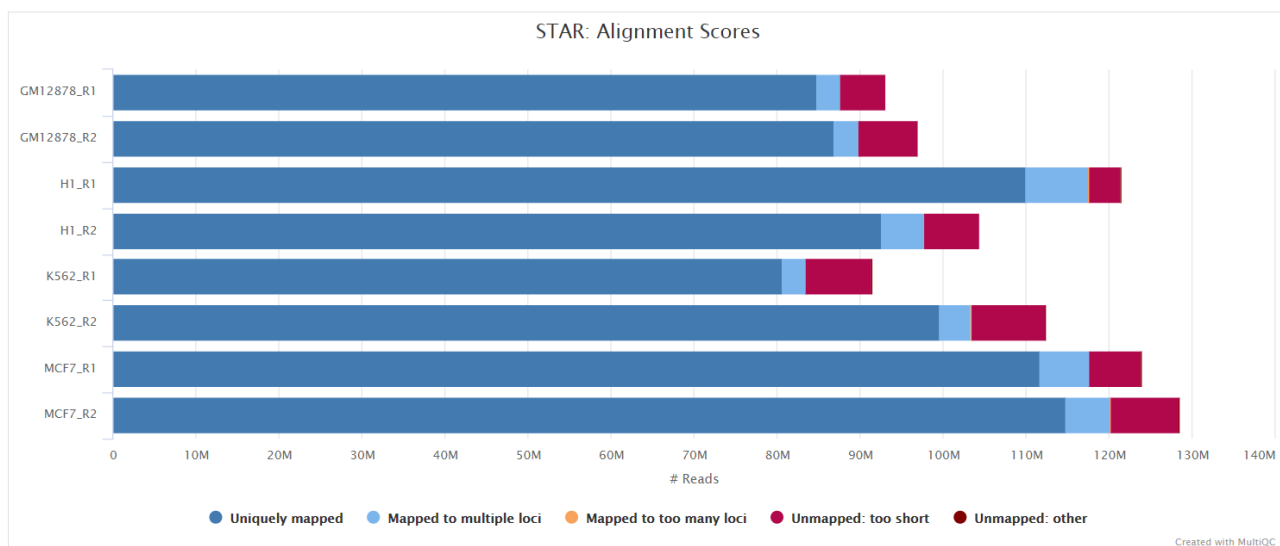
### STAR and Salmon

#### ► Output files

[STAR](#) is a read aligner designed for splice aware mapping typical of RNA sequencing data. STAR stands for *\*S\**pliced *\*T\**ranscripts *\*A\**lignment to a *\*R\**eference, and has been shown to have high accuracy and outperforms other aligners by more than a factor of 50 in mapping speed, but it is memory intensive. Using `--aligner star_salmon` is the default alignment and quantification option.

[Salmon](#) from [Ocean Genomics](#) is a tool for wicked-fast transcript quantification from RNA-seq data. It requires a set of target transcripts (either from a reference or de-novo assembly) in order to perform quantification. All you need to run Salmon is a FASTA file containing your reference transcripts and a set of FASTA/FASTQ/BAM file(s) containing your reads. The transcriptome-level BAM files generated by STAR are provided to Salmon for downstream quantification. You can of course also provide FASTQ files directly as input to Salmon in order to pseudo-align and quantify your data by providing the `--pseudo_aligner salmon` parameter. The results generated by the pipeline are exactly the same whether you provide BAM or FASTQ input so please see the [Salmon](#) results section for more details.

The STAR section of the MultiQC report shows a bar plot with alignment rates: good samples should have most reads as *Uniquely mapped* and few *Unmapped* reads.

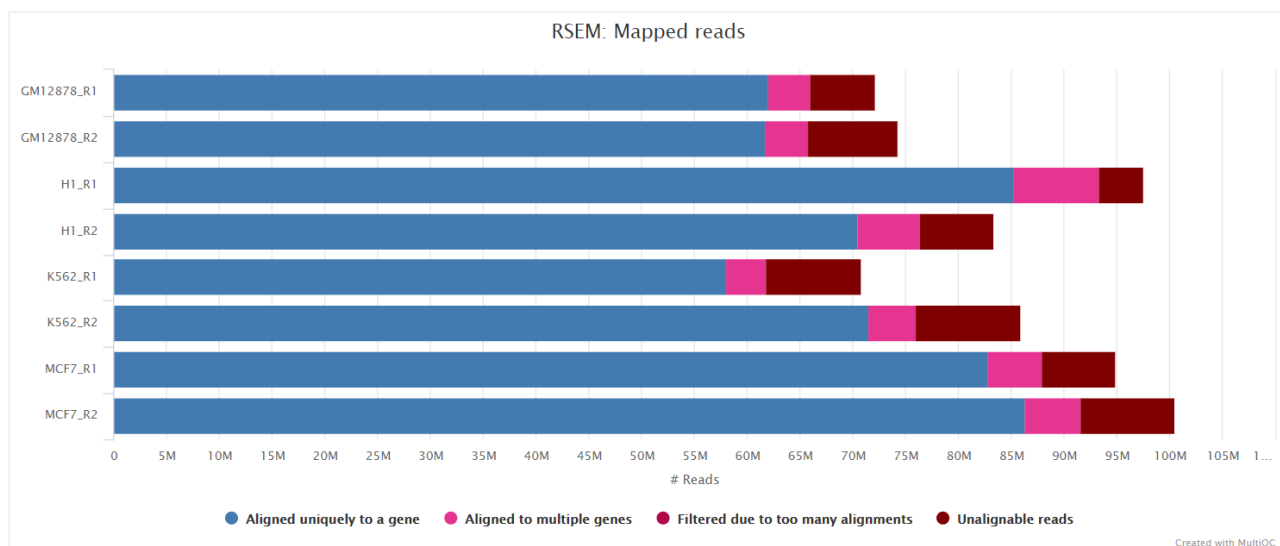


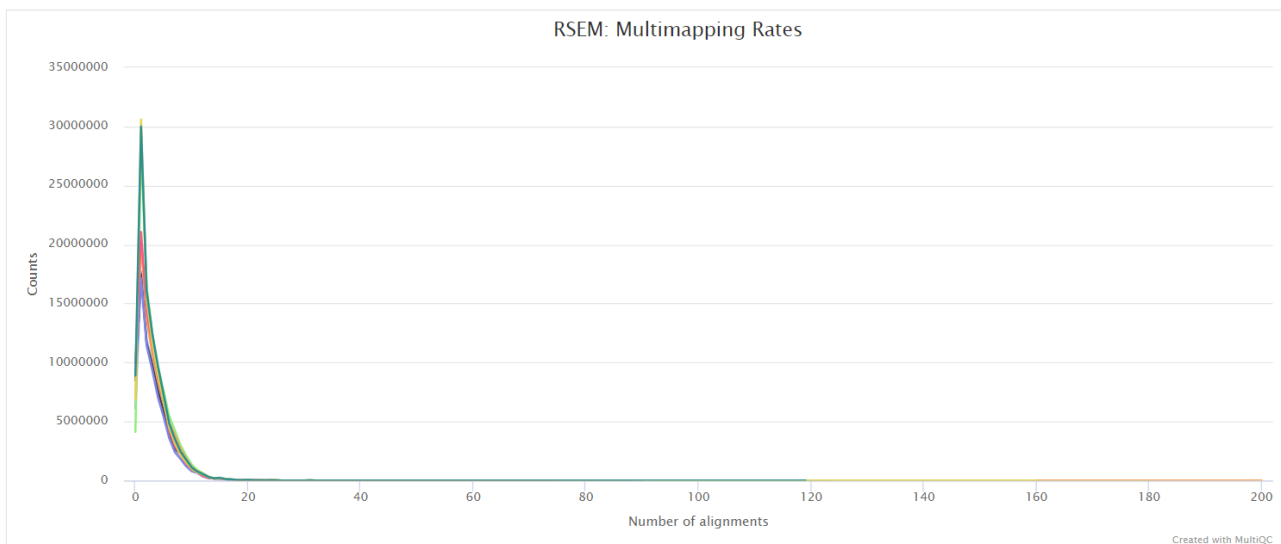
## STAR via RSEM

### ► Output files

[RSEM](#) is a software package for estimating gene and isoform expression levels from RNA-seq data. It has been widely touted as one of the most accurate quantification tools for RNA-seq analysis. RSEM wraps other popular tools to map the reads to the genome (i.e. STAR, Bowtie2, HISAT2; STAR is used in this pipeline) which are then subsequently filtered relative to a transcriptome before quantifying at the gene- and isoform-level. Other advantages of using RSEM are that it performs both the alignment and quantification in a single package and its ability to effectively use ambiguously-mapping reads.

You can choose to align and quantify your data with RSEM by providing the `--aligner star_rsem` parameter.



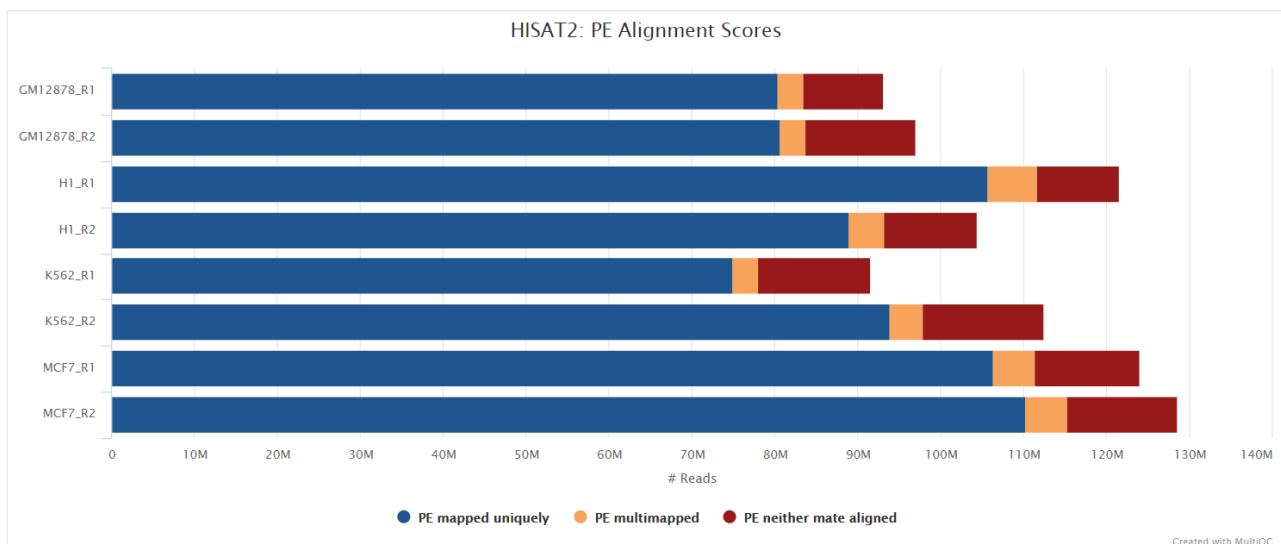


## HISAT2

### ► Output files

[HISAT2](#) is a fast and sensitive alignment program for mapping next-generation sequencing reads (both DNA and RNA) to a population of human genomes as well as to a single reference genome. It introduced a new indexing scheme called a Hierarchical Graph FM index (HGFM) which when combined with several alignment strategies, enable rapid and accurate alignment of sequencing reads. The HISAT2 route through the pipeline is a good option if you have memory limitations on your compute. However, quantification isn't performed if using `--aligner hisat2` due to the lack of an appropriate option to calculate accurate expression estimates from HISAT2 derived genomic alignments. However, you can use this route if you have a preference for the alignment, QC and other types of downstream analysis compatible with the output of HISAT2.

You can choose to align your data with HISAT2 by providing the `--aligner hisat2` parameter.



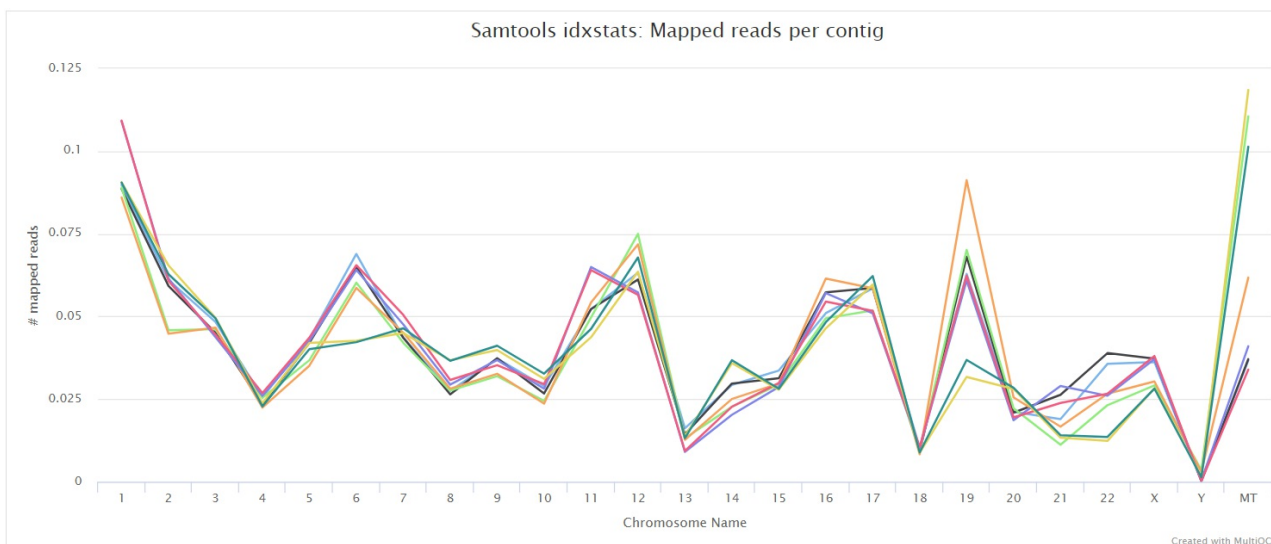
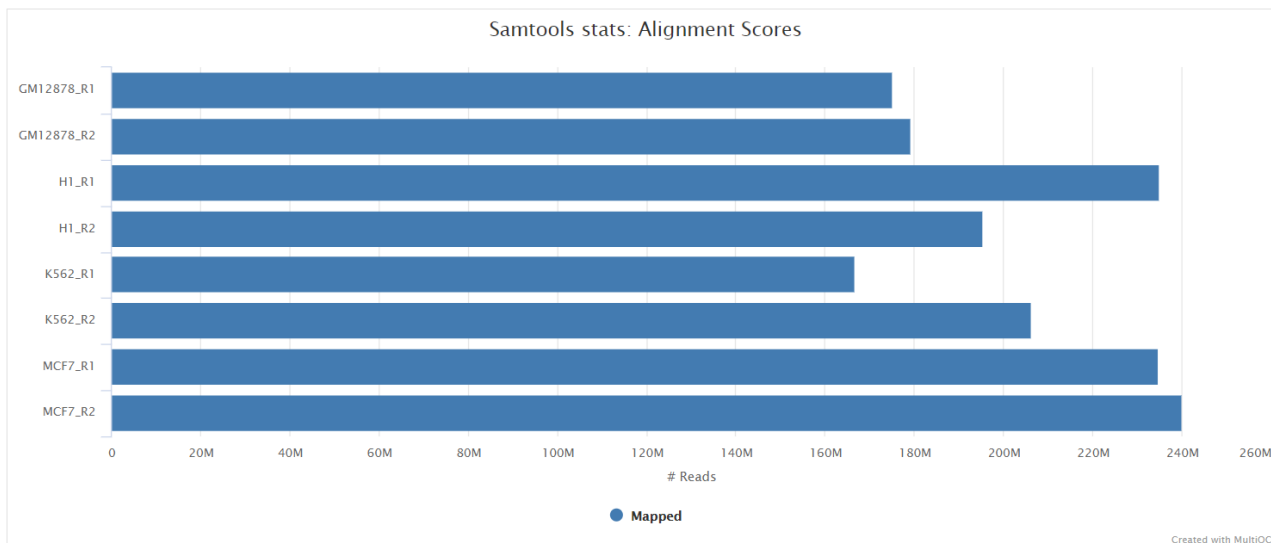
## Alignment post-processing

The pipeline has been written in a way where all the files generated downstream of the alignment are placed in the same directory as specified by `--aligner` e.g. if `--aligner star_salmon` is specified then all the downstream results will be placed in the `star_salmon/` directory. This helps with organising the directory structure and more importantly, allows the end-user to get the results from multiple aligners by simply re-running the pipeline with a different `--aligner` option along the `-resume` parameter. It also means that results won't be overwritten when resuming the pipeline and can be used for benchmarking between alignment algorithms if required.

## SAMtools

### ► Output files

The original BAM files generated by the selected alignment algorithm are further processed with [SAMtools](#) to sort them by coordinate, for indexing, as well as to generate read mapping statistics.



## UMI-tools dedup

### ► Output files

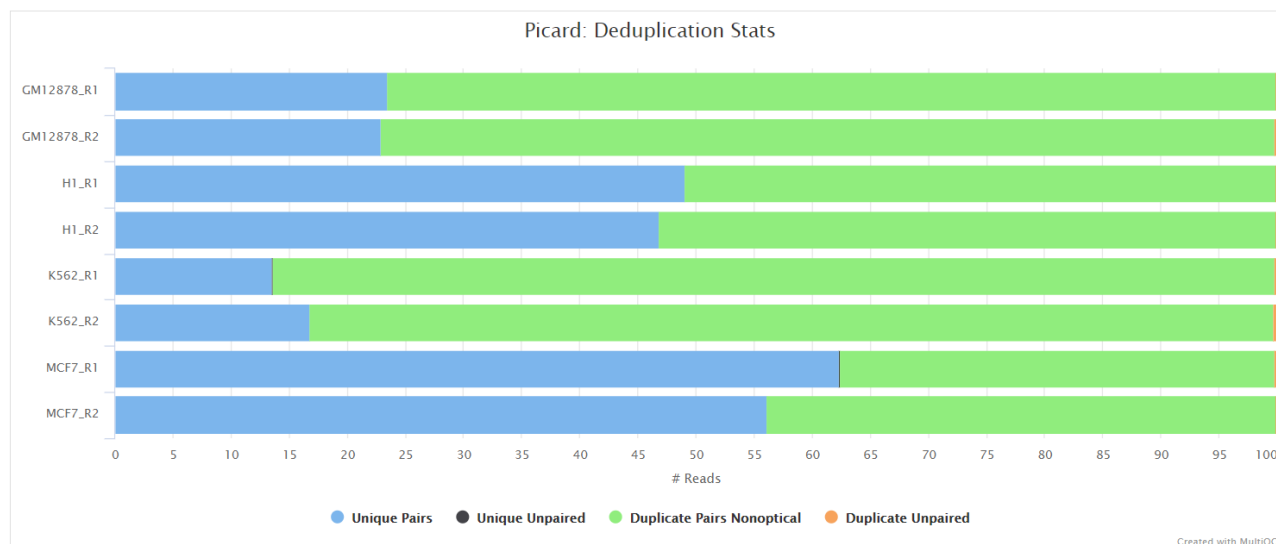
After extracting the UMI information from the read sequence (see [UMI-tools extract](#)), the second step in the removal of UMI barcodes involves deduplicating the reads based on both mapping and UMI barcode information using the UMI-tools `dedup` command. This will generate a filtered BAM file after the removal of PCR duplicates.

## picard MarkDuplicates

### ► Output files

Unless you are using [UMIs](#) it is not possible to establish whether the fragments you have sequenced from your sample were derived via true biological duplication (i.e. sequencing independent template fragments) or as a result of PCR biases introduced during the library preparation. By default, the pipeline uses [picard MarkDuplicates](#) to *mark* the duplicate reads identified amongst the alignments to allow you to gauge the overall level of duplication in your samples. However, for RNA-seq data it is not recommended to physically remove duplicate reads from the alignments (unless you are using UMIs) because you expect a significant level of true biological duplication that arises from the same fragments

being sequenced from for example highly expressed genes. This step will be skipped automatically when using the `--with_umi` option or explicitly via the `--skip_markduplicates` parameter.



## Other steps

### StringTie

#### ► Output files

[StringTie](#) is a fast and highly efficient assembler of RNA-Seq alignments into potential transcripts. It uses a novel network flow algorithm as well as an optional de novo assembly step to assemble and quantitate full-length transcripts representing multiple splice variants for each gene locus. In order to identify differentially expressed genes between experiments, StringTie's output can be processed by specialized software like [Ballgown](#), [Cuffdiff](#) or other programs ([DESeq2](#), [edgeR](#), etc.).

### BEDTools and bedGraphToBigWig

#### ► Output files

The [bigWig](#) format is an indexed binary format useful for displaying dense, continuous data in Genome Browsers such as the [UCSC](#) and [IGV](#). This mitigates the need to load the much larger BAM files for data visualisation purposes which will be slower and result in memory issues. The bigWig format is also supported by various bioinformatics software for downstream processing such as meta-profile plotting.

## Quality control

### RSeQC

[RSeQC](#) is a package of scripts designed to evaluate the quality of RNA-seq data. This pipeline runs several, but not all RSeQC scripts. You can tweak the supported scripts you would like to run by adjusting the `--rseqc_modules` parameter which by default will run all of the following: `bam_stat.py`, `inner_distance.py`, `infer_experiment.py`, `junction_annotation.py`, `junction_saturation.py`, `read_distribution.py` and `read_duplication.py`.

The majority of RSeQC scripts generate output files which can be plotted and summarised in the MultiQC report.

#### Infer experiment

#### ► Output files

This script predicts the "strandedness" of the protocol (i.e. unstranded, sense or antisense) that was used to prepare the sample for sequencing by assessing the

orientation in which aligned reads overlay gene features in the reference genome. The strandedness of each sample has to be provided to the pipeline in the input samplesheet (see [usage docs](#)). However, this information is not always available, especially for public datasets. As a result, additional features have been incorporated into this pipeline to auto-detect whether you have provided the correct information in the samplesheet, and if this is not the case then a warning table will be placed at the top of the MultiQC report highlighting the offending samples (see image below). If required, this will allow you to correct the input samplesheet and rerun the pipeline with the accurate strand information. Note, it is important to get this information right because it can affect the final results.

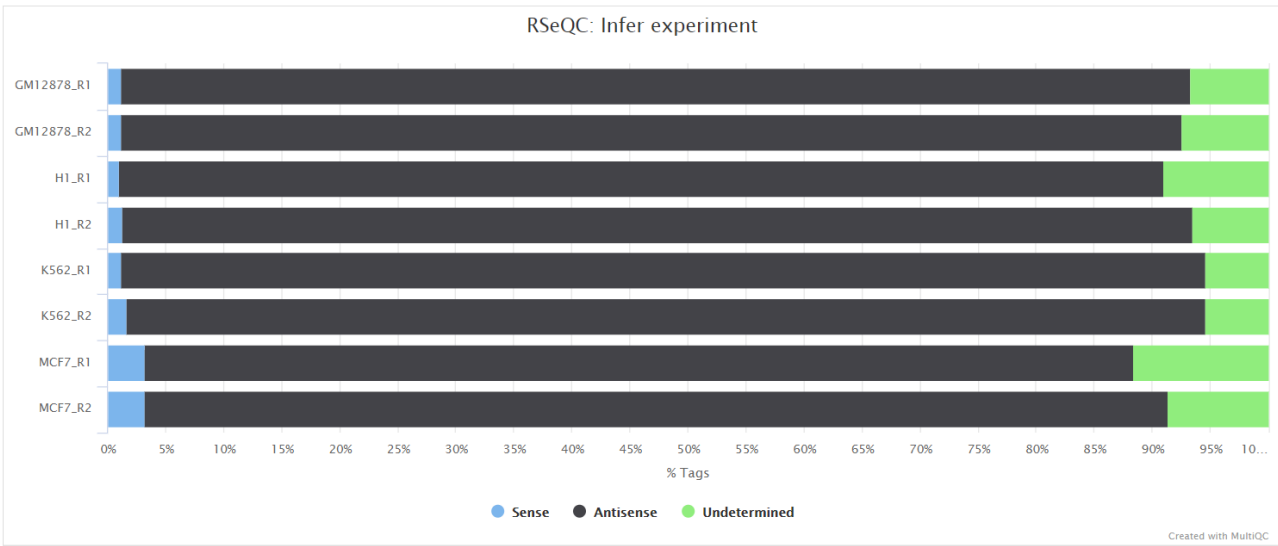
RSeQC documentation: [infer\\_experiment.py](#)

## WARNING: Fail Strand Check

List of samples that failed the strandedness check between that provided in the samplesheet and calculated by the RSeQC [infer\\_experiment.py](#) tool.

Copy tableConfigure ColumnsPlotShowing 2/2 rows and 5/5 columns.

Sample	Provided strandedness	Inferred strandedness	Sense (%)	Antisense (%)	Undetermined (%)
RAP1_IAA_30M_R1	unstranded	reverse	1.26	83.04	15.70
WT_R1	forward	reverse	1.58	85.09	13.33

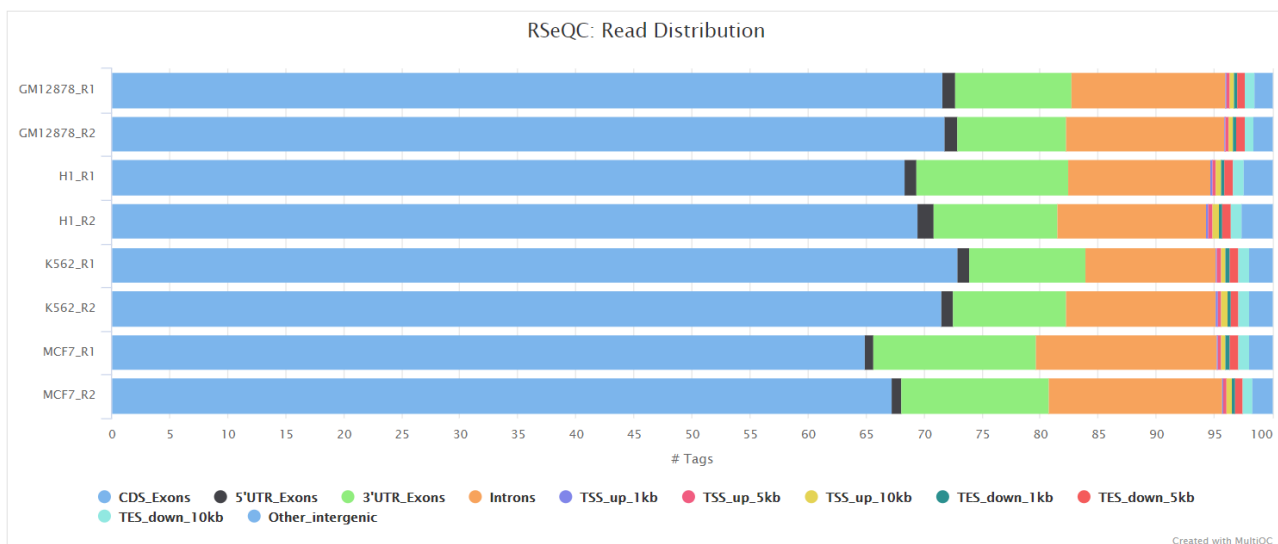


## Read distribution

### ► Output files

This tool calculates how mapped reads are distributed over genomic features. A good result for a standard RNA-seq experiments is generally to have as many exonic reads as possible ([CDS Exons](#)). A large amount of intronic reads could be indicative of DNA contamination in your sample but may be expected for a total RNA preparation.

RSeQC documentation: [read\\_distribution.py](#)

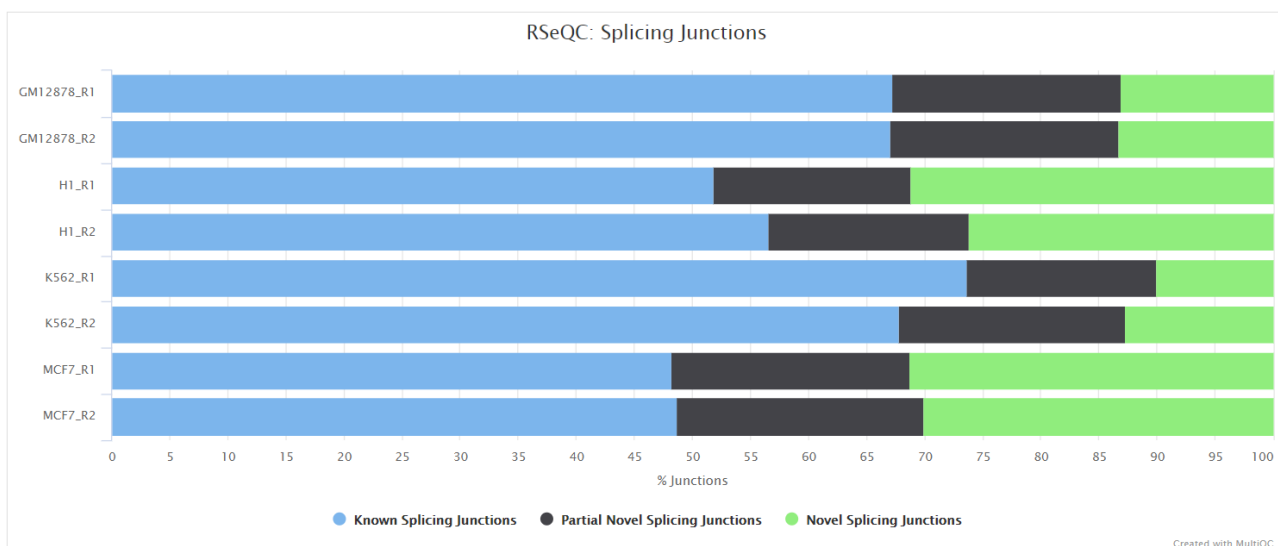


## Junction annotation

### ► Output files

Junction annotation compares detected splice junctions to a reference gene model. Splicing annotation is performed in two levels: splice event level and splice junction level.

RSeQC documentation: [junction\\_annotation.py](#)



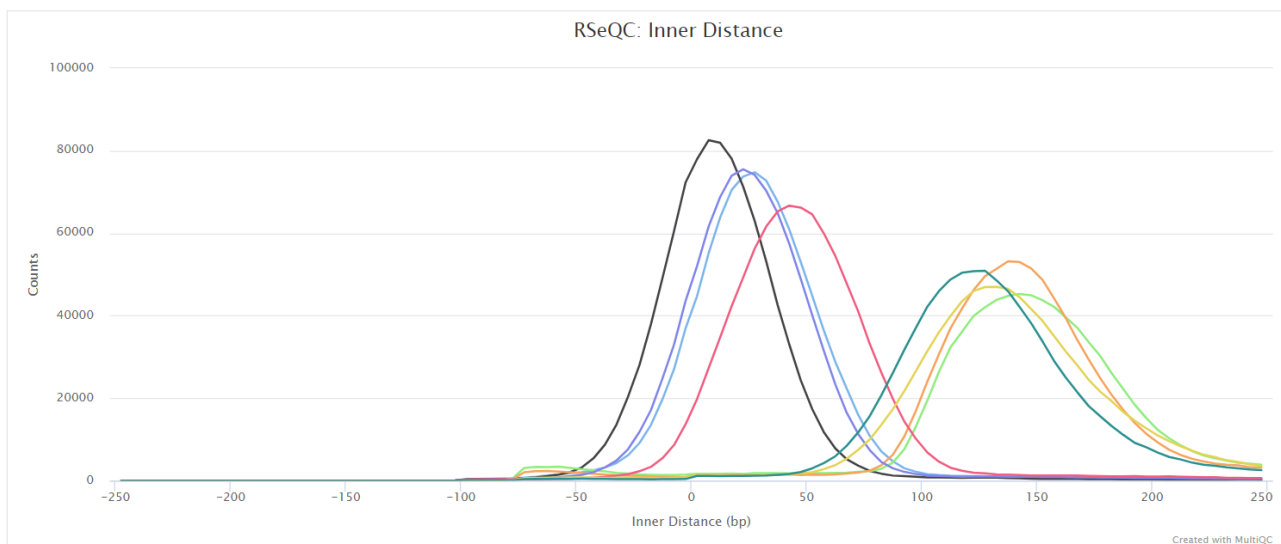
## Inner distance

### ► Output files

The inner distance script tries to calculate the inner distance between two paired-end reads. It is the distance between the end of read 1 to the start of read 2, and it is sometimes confused with the insert size (see [this blog post](#) for disambiguation):

This plot will not be generated for single-end data. Very short inner distances are often seen in old or degraded samples (eg. FFPE) and values can be negative if the reads overlap consistently.

RSeQC documentation: [inner\\_distance.py](#)

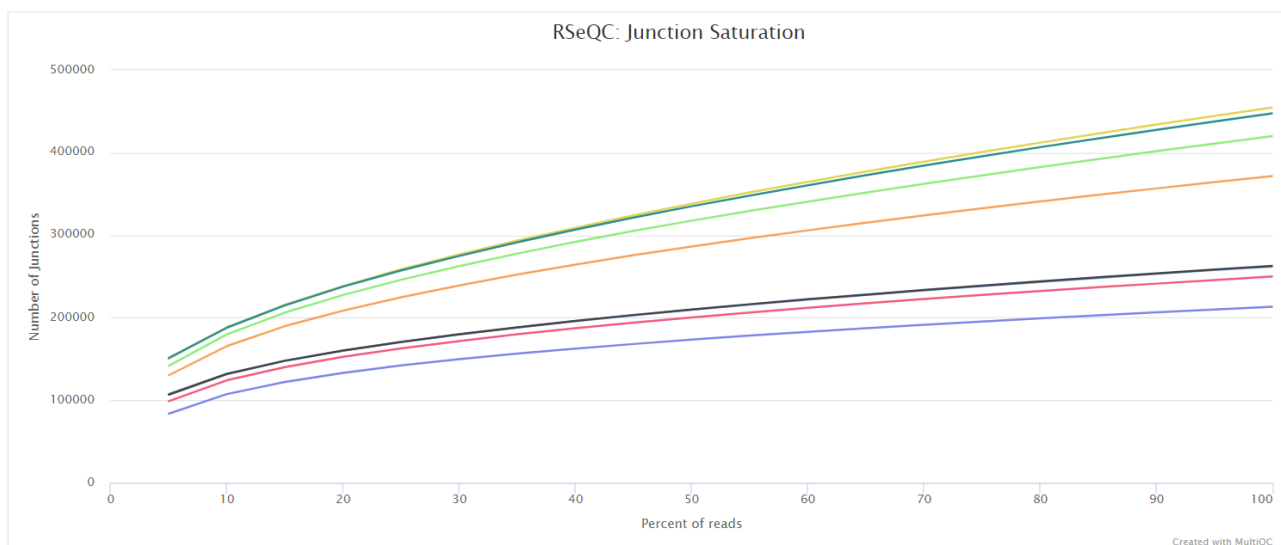


## Junction saturation

### ► Output files

This script shows the number of splice sites detected within the data at various levels of subsampling. A sample that reaches a plateau before getting to 100% data indicates that all junctions in the library have been detected, and that further sequencing will not yield any more observations. A good sample should approach such a plateau of *Known junctions*, however, very deep sequencing is typically required to saturate all *Novel Junctions* in a sample.

RSeQC documentation: [junction\\_saturation.py](#)

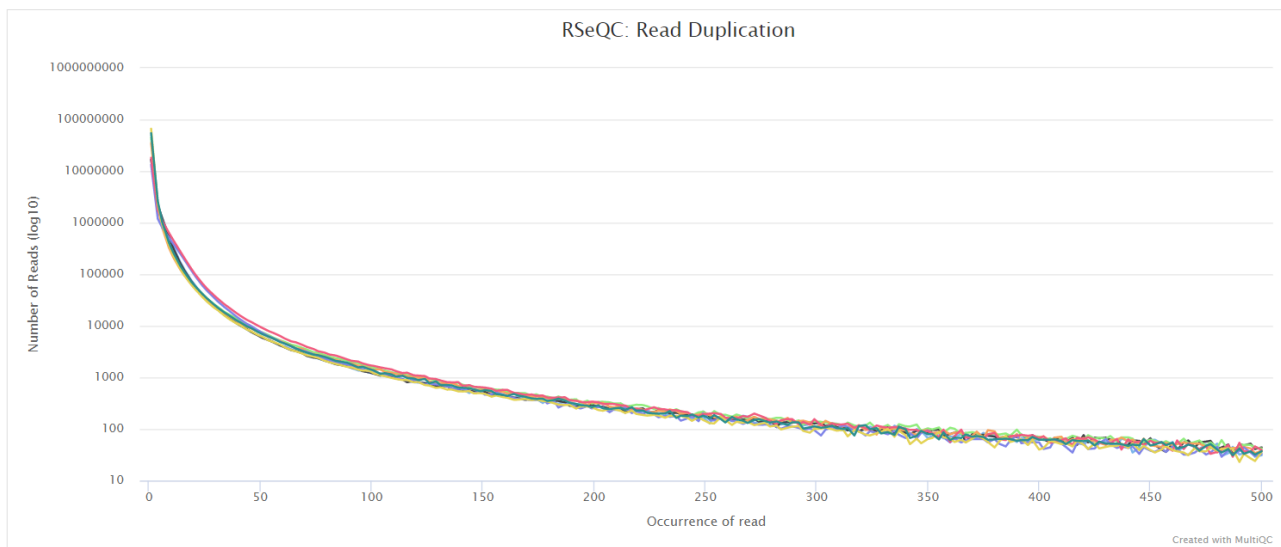


## Read duplication

### ► Output files

This plot shows the number of reads (y-axis) with a given number of exact duplicates (x-axis). Most reads in an RNA-seq library should have a low number of exact duplicates. Samples which have many reads with many duplicates (a large area under the curve) may be suffering excessive technical duplication.

RSeQC documentation: [read\\_duplication.py](#)



## BAM stat

### ► Output files

This script gives numerous statistics about the aligned BAM files. A typical output looks as follows:

```
#Output (all numbers are read count)
#=====
Total records:                41465027
QC failed:                    0
Optical/PCR duplicate:       0
Non Primary Hits              8720455
Unmapped reads:              0

mapq < mapq_cut (non-unique): 3127757
mapq >= mapq_cut (unique):    29616815
Read-1:                      14841738
Read-2:                      14775077
Reads map to '+':            14805391
Reads map to '-':            14811424
Non-splice reads:            25455360
Splice reads:                4161455
Reads mapped in proper pairs: 21856264
Proper-paired reads map to different chrom: 7648
```

MultiQC plots each of these statistics in a dot plot. Each sample in the project is a dot - hover to see the sample highlighted across all fields.

RSeQC documentation: [bam\\_stat.py](#)

## TIN

### ► Output files

This script is designed to evaluate RNA integrity at the transcript level. TIN (transcript integrity number) is named in analogous to RIN (RNA integrity number). RIN (RNA integrity number) is the most widely used metric to evaluate RNA integrity at sample (or transcriptome) level. It is a very useful preventive measure to ensure good RNA quality and robust, reproducible RNA sequencing. This process isn't run by default - please see [this issue](#).

RSeQC documentation: [tin.py](#)

## Qualimap

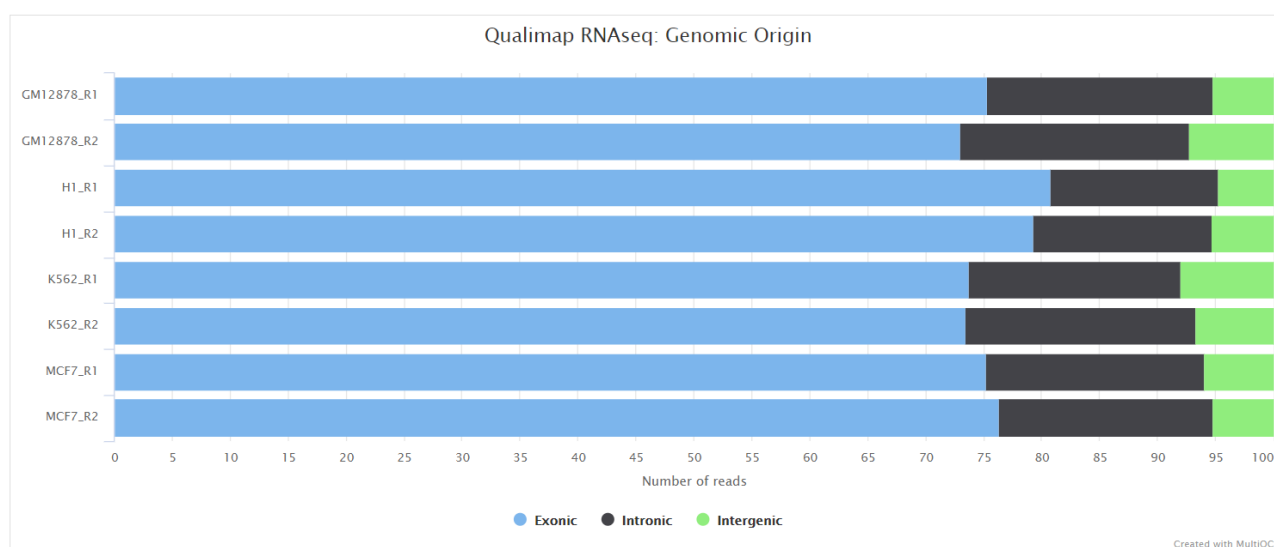
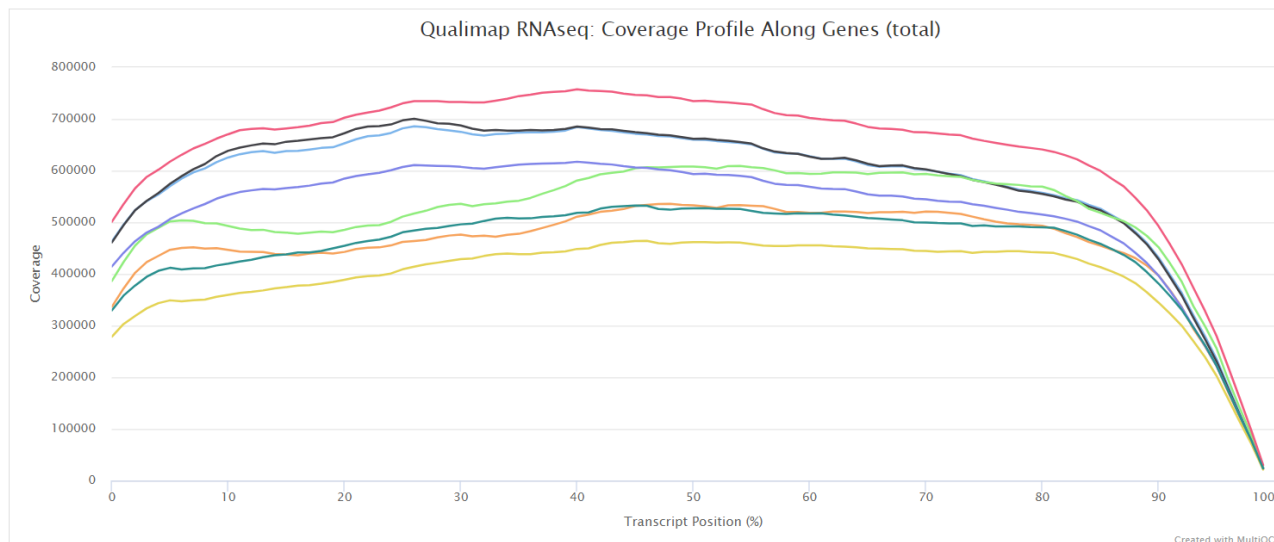
### ► Output files

[Qualimap](#) is a platform-independent application written in Java and R that

provides both a Graphical User Interface (GUI) and a command-line interface to facilitate the quality control of alignment sequencing data. Shortly, Qualimap:

- Examines sequencing alignment data according to the features of the mapped reads and their genomic properties.
- Provides an overall view of the data that helps to detect biases in the sequencing and/or mapping of the data and eases decision-making for further analysis.

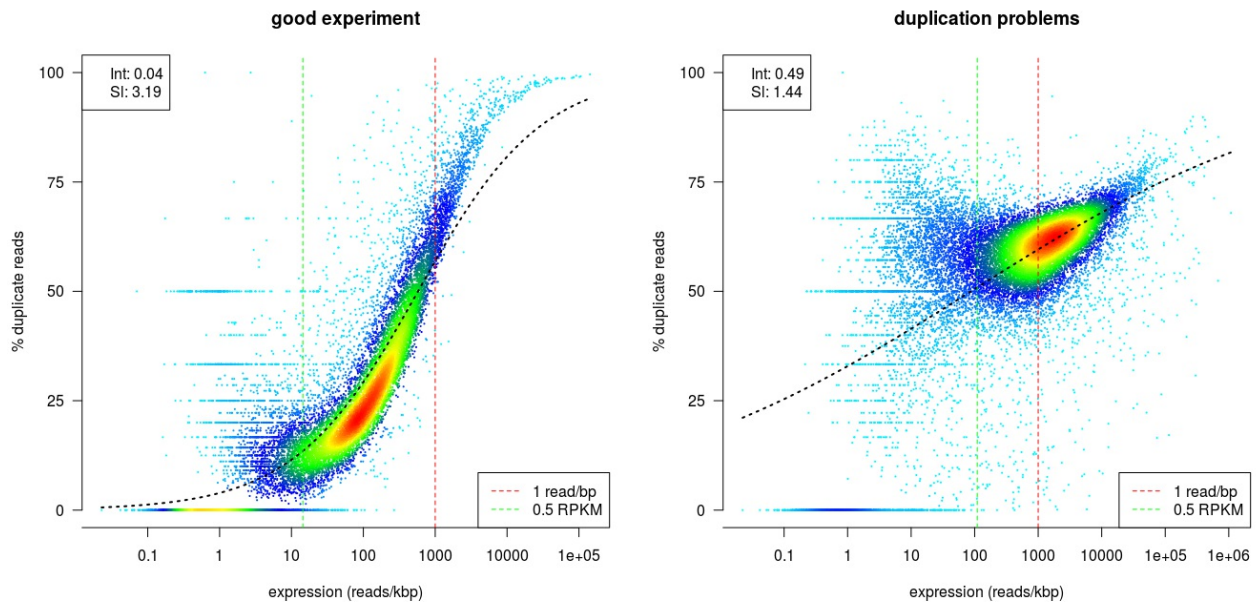
The [Qualimap RNA-seq QC module](#) is used within this pipeline to assess the overall mapping and coverage relative to gene features.



## dupRadar

### ► Output files

[dupRadar](#) is a Bioconductor library written in the R programming language. It generates various QC metrics and plots that relate duplication rate with gene expression levels in order to identify experiments with high technical duplication. A good sample with little technical duplication will only show high numbers of duplicates for highly expressed genes. Samples with technical duplication will have high duplication for all genes, irrespective of transcription level.

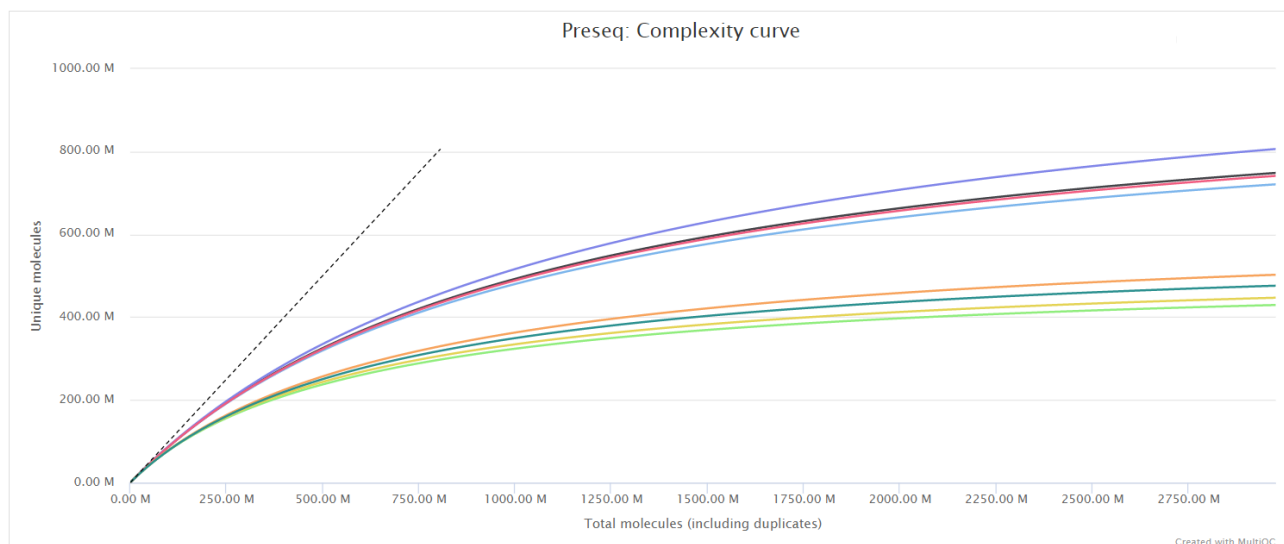


Credit: [dupRadar documentation](#)

## Preseq

### ► Output files

The [Preseq](#) package is aimed at predicting and estimating the complexity of a genomic sequencing library, equivalent to predicting and estimating the number of redundant reads from a given sequencing depth and how many will be expected from additional sequencing using an initial sequencing experiment. The estimates can then be used to examine the utility of further sequencing, optimize the sequencing depth, or to screen multiple libraries to avoid low complexity samples. A shallow curve indicates that the library has reached complexity saturation and further sequencing would likely not add further unique reads. The dashed line shows a perfectly complex library where total reads = unique reads. Note that these are predictive numbers only, not absolute. The MultiQC plot can sometimes give extreme sequencing depth on the X axis - click and drag from the left side of the plot to zoom in on more realistic numbers.

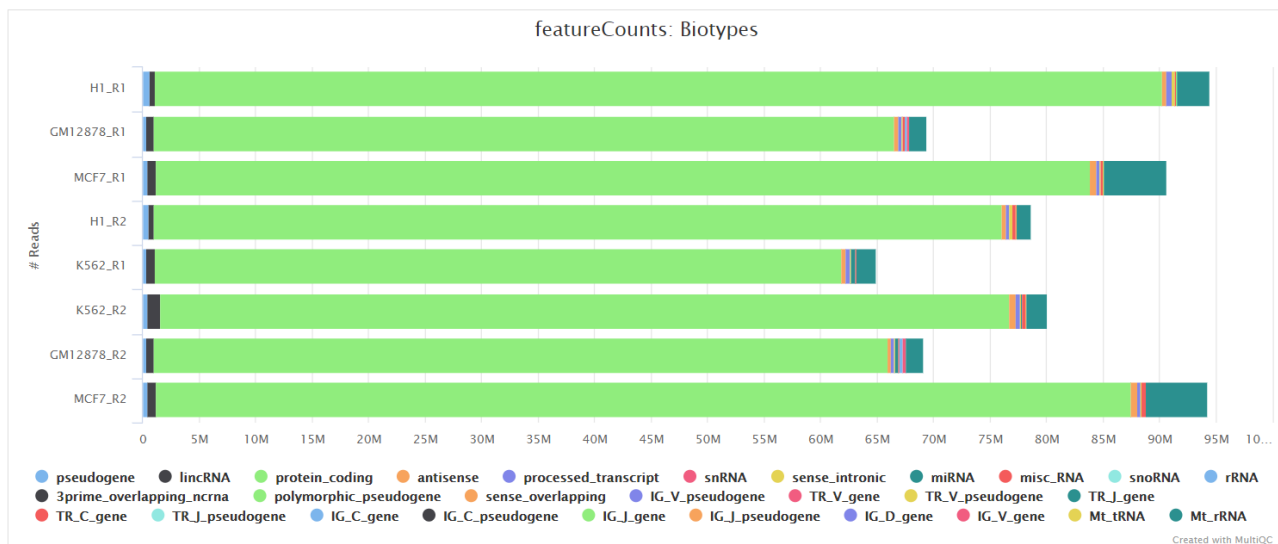


## featureCounts

### ► Output files

[featureCounts](#) from the [Subread](#) package is a quantification tool used to summarise the mapped read distribution over genomic features such as genes, exons, promoters, gene bodies, genomic bins and chromosomal locations. We can also use featureCounts to count overlaps with different classes of genomic

features. This provides an additional QC to check which features are most abundant in the sample, and to highlight potential problems such as rRNA contamination.



## DESeq2

### ► Output files

[DESeq2](#) is one of the most commonly used software packages to perform differential expression analysis for RNA-seq datasets.

**This pipeline uses a standardised DESeq2 analysis script to get an idea of the reproducibility across samples within the experiment. Please note that this will not suit every experimental design, and if there are other problems with the experiment then it may not work as well as expected.**

The script included in the pipeline uses DESeq2 to normalise read counts across all of the provided samples in order to create a PCA plot and a clustered heatmap showing pairwise Euclidean distances between the samples in the experiment. These help to show the similarity between groups of samples and can reveal batch effects and other potential issues with the experiment.

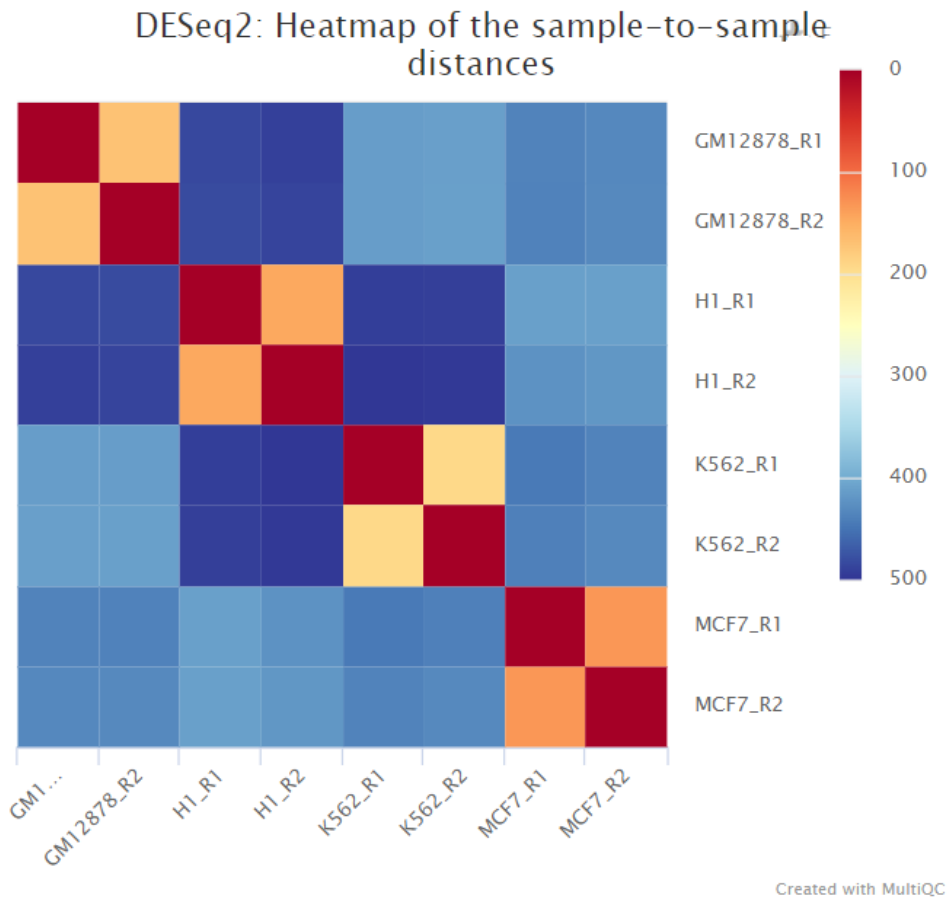
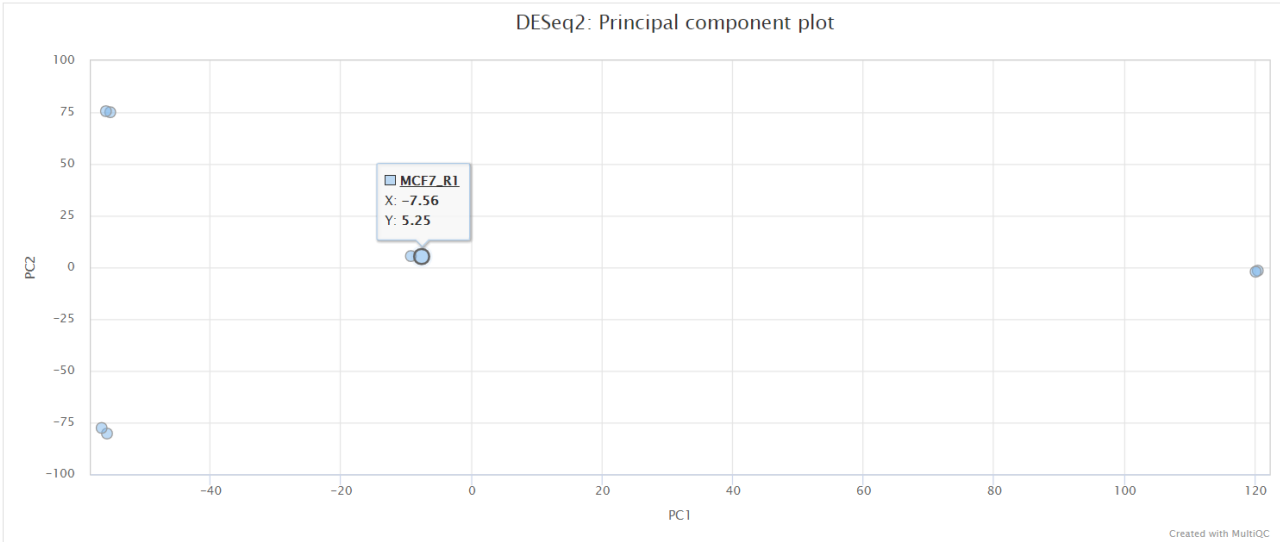
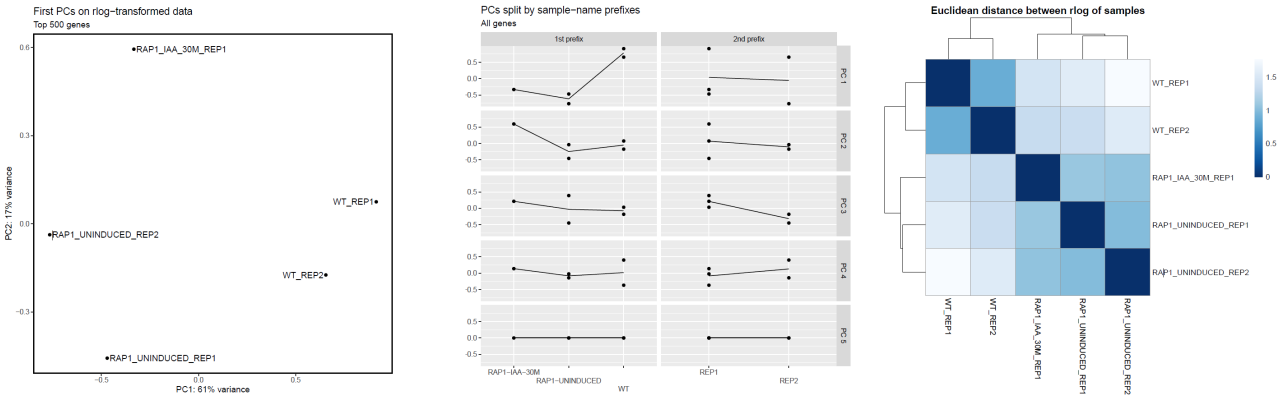
By default, the pipeline uses the `vst` transformation which is more suited to larger experiments. You can set the parameter `--deseq2_vst false` if you wish to use the DESeq2 native `rlog` option. See [DESeq2 docs](#) for a more detailed explanation.

The PCA plots are generated based alternately on the top five hundred most variable genes, or all genes. The former is the conventional approach that is more likely to pick up strong effects (ie the biological signal) and the latter, when different, is picking up a weaker but consistent effect that is synchronised across many transcripts. We project both of these onto the first two PCs (shown in the top row of the figure below), which is the best two dimensional representation of the variation between samples.

We also explore higher components in terms of experimental factors inferred from sample names. If your sample naming convention follows a strict policy of using underscores to delimit values of experimental factors (for example `WT_UNTREATED_REP1`) and all names have the same number of underscores (so excluding `WT_TREATED_10mL_REP1` from being compatible with the previous label), then any of these factors that are informative (ie label some but not all samples the same) then we individually plot upto the first five PCs, per experimental level, for each of the experimental factors.

The plot on the left hand side shows the standard PC plot - notice the variable number of underscores, meaning that the central plot would not be produced: here we have changed the underscore that is hyphenating the treatment to a '-' character. This allows the central plot to be generated, and we can see that replicate (the 2nd part of the sample name) seems to be affecting the 3rd principal component, but the treatment factor is affecting the more important first

two components. The right-most plot shows all pairwise euclidean distances between the samples.



## ► Output files

[MultiQC](#) is a visualization tool that generates a single HTML report summarising all samples in your project. Most of the pipeline QC results are visualised in the report and further statistics are available in the report data directory.

Results generated by MultiQC collate pipeline QC from supported tools i.e. FastQC, Cutadapt, SortMeRNA, STAR, RSEM, HISAT2, Salmon, SAMtools, Picard, RSeQC, Qualimap, Preseq and featureCounts. Additionally, various custom content has been added to the report to assess the output of dupRadar, DESeq2 and featureCounts biotypes, and to highlight samples failing a minimum mapping threshold or those that failed to match the strand-specificity provided in the input samplesheet. The pipeline has special steps which also allow the software versions to be reported in the MultiQC output for future traceability. For more information about how to use MultiQC reports, see <http://multiqc.info>.

# Pseudo-alignment and quantification

## Salmon

## ► Output files

As described in the [STAR and Salmon](#) section, you can choose to pseudo-align and quantify your data with [Salmon](#) by providing the `--pseudo-aligner salmon` parameter. By default, Salmon is run in addition to the standard alignment workflow defined by `--aligner`, mainly because it allows you to obtain QC metrics with respect to the genomic alignments. However, you can provide the `--skip_alignment` parameter if you would like to run Salmon in isolation. If Salmon is run in isolation, the outputs mentioned above will be found in a folder named `salmon`. If Salmon is run alongside STAR, the folder will be named `star_salmon`.

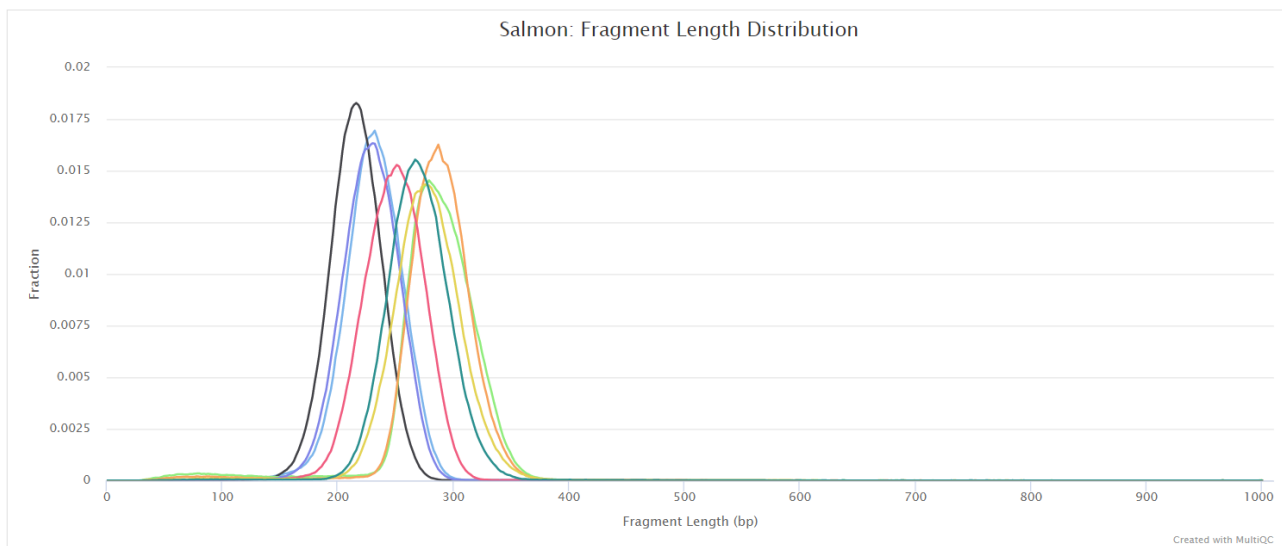
Transcripts with large inferential uncertainty won't be assigned the exact number of reads reproducibly, every time Salmon is run. Read more about this on the [nf-core/maseq](#) and [salmon](#) Github repos.

The [tximport](#) package is used in this pipeline to summarise the results generated by Salmon into matrices for use with downstream differential analysis packages. We use tximport with different options to summarize count and TPM quantifications at the gene- and transcript-level. Please see [#499](#) for discussion and links regarding which counts are suitable for different types of analysis.

According to the `tximport` documentation you can do one of the following:

- Use bias corrected counts with an offset: import all the salmon files with `tximport` and then use `DESeq2` with `dds <- DESeqDataSetFromTximport(txi, sampleTable, ~condition)` to correct for changes to the average transcript length across samples.
- Use bias corrected counts without an offset: load and use `salmon.merged.gene_counts_length_scaled.tsv` or `salmon.merged.gene_counts_scaled.tsv` directly as you would with a regular counts matrix.
- Use bias uncorrected counts: load and use the `txi$counts` matrix (or `salmon.merged.gene_counts.tsv`) with `DESeq2`. This does not correct for potential differential isoform usage. Alternatively, if you have 3' tagged RNA-seq data this is the most suitable method.

**NB:** The default Salmon parameters and a k-mer size of 31 are used to create the index. As [documented here](#) and [discussed here](#), a k-mer size off 31 works well with reads that are 75bp or longer.



## Workflow reporting and genomes

### Reference genome files

#### ► Output files

A number of genome-specific files are generated by the pipeline because they are required for the downstream processing of the results. If the `--save_reference` parameter is provided then these will be saved in the `genome/` directory. It is recommended to use the `--save_reference` parameter if you are using the pipeline to build new indices so that you can save them somewhere locally. The index building step can be quite a time-consuming process and it permits their reuse for future runs of the pipeline to save disk space.

### Pipeline information

#### ► Output files

[Nextflow](#) provides excellent functionality for generating various reports relevant to the running and execution of the pipeline. This will allow you to troubleshoot errors with the running of the pipeline, and also provide you with other information such as launch commands, run times and resource usage.