**- Notes -**

**RNAseq data analysis**

***By Pawel Rosikiewicz***

***www.SimpleAI.ch***

## **Data Pre-processing**

The read counts provided for differential expression analysis need to be normalised, before giving them as input to DeSeq or edgeR(differential expression analysis tools in R). The normalisation techniques are :

### **Log Transform -**

The log of raw read counts is taken which results in increasing the distance between small measurements and decreasing the distance between large measurements.

### **Quantile Normalisation -**

Multi-sample normalization techniques such as quantile normalization have become a standard and essential part of analysis pipelines for high-throughput data.These techniques transform the original raw data to remove unwanted technical variation. Technical variation can cause perceived differences between samples processed on high-throughput technologies, irrespective of the biological variation. These differences are typically due to changes in experimental conditions that are hard or impossible to control and confusing them with biological variability can lead to false discoveries.

### **Filter -**

The genes with relatively low or no expression value are filtered out from the final normalised read count dataset which is then given as input to DeSeq or edgeR.

Chart, box and whisker chart

Description automatically generated

### **Technical Batch Effects**

Technical batch effect occurs when non-biological factors in an experiment cause changes in the data produced by the experiment. Such effects can lead to inaccurate conclusions when their causes are correlated with one or more outcomes of interest in an experiment. Thus, we need to balance the variables of our interest in an experiment in order to remove the batch effects. Techniques involved are :

1. PCA(Principal Component Analysis)
2. MDS(Multidimensional Scaling)

These techniques reduce the representation of each sample from a vector of thousands of measurements to a vector of length of number of samples. Also, this vector captures the largest sources of variation in the dataset. The unwanted variations, which interfere with the variable of interest are then removed, thus eliminating the batch effect.

Chart, scatter chart

Description automatically generated

### **Methods for Differential Expression Analysis**

There are different pipelines for differential expression analysis in R such as **edgeR** and **DESeq** based on **negative binomial (NB) distributions**. It is important to consider the experimental design when choosing an analysis method. While some of the differential expression tools can only perform **pairwise comparison,** others such as **edgeR, limma-voom, DESeq can perform multiple comparisons**.

**These tools basically follow the same approach, i.e. ,**

* they estimate the gene expression difference for a given gene,
* followed by a statistical test based on the null hypothesis that the difference is close to zero, which would mean that there is no difference in the gene expression values that could be explained by the conditions.
* These tools are all based on the **R language** and make heavy use of numerous statistical methods that have been developed and implemented over the past two decades to improve the power to detect robust changes based on extremely small numbers of replicates. Estimating the difference between read counts :

**DESeq2 and edgeR**

* rely on a negative binomial model to fit the observed read counts to arrive at the estimate for the difference.
* Originally, read counts had been modeled using the Poisson distribution because:

**Why Negative Binomial ?**

* individual reads can be interpreted as binary data (Bernoulli trials): they either originate from a single gene A or not.
* we are trying to model the discrete probability distribution of the number of successes (success = read is present in the sequenced library).
* the pool of possible reads that could be present is large, while the proportion of reads belonging to gene A is quite small.

**Why Poisson distribution ?**

* The convenient feature of a Poisson distribution is that **variance = mean**. Thus, if the RNA-seq experiment gives us a precise estimate of the mean read counts per condition, we implicitly know what kind of variance to expect for read counts that are not truly changing between two conditions. This, in turn, then allows us to identify those genes that show greater differences between the two conditions than expected by chance.

**So why today we are using Negative Binomial ?**

* Unfortunately, **only read counts of the same library preparation (= technical replicates) can be well approximated by the Poisson distribution**, biological replicates have been shown to display greater variance (noise).
* **This overdispersion can be captured with the negative binomial distribution**, which is a more general form of the Poisson distribution where the variance is allowed to exceed the mean
* This means that we now need to estimate two parameters from the read counts: **the mean as well as the dispersion.** The precision of these estimates strongly depends on the number (and variation) of replicates – the more replicates, the better the grasp on the underlying mean expression values of unchanged genes and the variance that is due to biological variation rather than the experimental treatment.

### **Testing the null hypothesis :**

* **The null hypothesis** is that there is no systematic difference between the average read count values of the different conditions for a given gene.
* **The p-values** are assigned by these tools using some variation of the well-known t−test (How dissimilar are the means of two populations?) or ANOVAs (How well does a reduced model capture the data when compared to the full model with all coefficients?).
* Once a list of p-values for all the genes of our data set is obtained, it is important to realize that the same type of test has been performed for thousands and thousands of genes. That means, that even if genes with a p-value smaller than 0.05 are considered, and there are 1000 genes, there may be 0.05∗1000 = 50 false positive hits. Consequently, all the tools offer some sort of **correction for this multiple testing hypotheses like Benjamini-Hochberg formula**.

### **RAseq data analysis - general Methodology**

1. Raw reads (**FASTQ files**) undergo **quality assessment** and filtering.
2. The quality-filtered reads are **aligned to the reference genome** via aligners like **HiSat or TopHat2**
3. The mapped reads are **summarised and aggregated over genes via HTSeq**.
4. For baseline expression, the **FPKMsn (Fragments Per Kilobase Million)** are calculated from the raw counts by **iRAP**.
5. These are **averaged for each set of technical replicates**,
6. and then **quantile normalised within each set of biological replicates** using limma.
7. Finally, they are **averaged for all biological replicates** (if any).
8. For differential expression, **genes expressed differentially between the test and the reference** groups of each **pairwise** contrast are identified using **DESeq2**.

### **Biological interpretation of gene expression data**

1. A common method of visualising gene expression data is to display it as a heatmap (Figure 12). The heatmap may also be combined with clustering methods which group genes and/or samples together based on the similarity of their gene expression pattern. This can be useful for identifying genes that are commonly regulated, upregulated or downregulated(based on log2 fold change values), or biological signatures associated with a particular condition (e.g a disease or an environmental condition).

Chart, timeline

Description automatically generated

1. A common approach to interpreting gene expression data is **gene set enrichment analysis** **based on** **the functional annotation of the differentially expressed genes**. This is useful for finding out if the differentially expressed genes are associated with a certain biological process or molecular function.
2. **The Gene Ontology**, containing standardised annotation of gene products, is commonly used for this purpose. It works by comparing the frequency of individual annotations in the gene list (e.g differentially expressed genes) with a reference list (usually all genes on the microarray or in the genome). Enrichment of biological pathways supplied by **KEGG, Ingenuity, Reactome or WikiPathways** can be performed in a similar way.

### **Reference:**

* <https://www.ebi.ac.uk/training/online/course/functional-genomics-ii-common-technologies-and-data-analysis-methods/differential-gene>
* <https://www.datacamp.com/courses/differential-expression-analysis-with-limma-in-r>
* <https://chagall.med.cornell.edu/RNASEQcourse/Intro2RNAseq.pdf>
* <https://en.wikipedia.org/wiki/Batch_effect>

# RPKM, FPKM and TPM

From : <https://www.rna-seqblog.com/rpkm-fpkm-and-tpm-clearly-explained/>

It used to be when you did RNA-seq, you reported your results in **RPKM (Reads Per Kilobase Million)** or **FPKM (Fragments Per Kilobase Million).** However, **TPM (Transcripts Per Kilobase Million**) is now becoming quite popular. Since there seems to be a lot of confusion about these terms, I thought I’d use a StatQuest to clear everything up.

**These three metrics attempt to normalize for sequencing depth and gene length.**

**RPKM (Reads Per Kilobase Million)**

Here’s how you do it for RPKM: - RPKM was made for single-end RNA-seq

1. **Count up the total reads in a sample** and divide that number by 1,000,000 – this is our “per million” scaling factor.
2. **Divide the read counts by the “per million” scaling factor**. This normalizes for sequencing depth, giving you reads per million (RPM)
3. **Divide the RPM values by the length of the gene, in kilobases**. This gives you RPKM.

**FPKM (Fragments Per Kilobase Million).**

**FPKM is very similar to RPKM**. RPKM was made for single-end RNA-seq, where every read corresponded to a single fragment that was sequenced. FPKM was **made for paired-end RNA-seq**. With paired-end RNA-seq, two reads can correspond to a single fragment, or, if one read in the pair did not map, one read can correspond to a single fragment. The only difference between RPKM and FPKM is that FPKM **takes into account that two reads can map to one fragment** (and so it doesn’t count this fragment twice).

**TPM (Transcripts Per Kilobase Million**)

TPM is very similar to RPKM and FPKM. The only difference is the order of operations. Here’s how you calculate TPM:

1. **Divide the read counts by the length of each gene in kilobases**. This gives you reads per kilobase (RPK).
2. Count up all the RPK values in a sample and divide this number by 1,000,000. This is your “per million” scaling factor.
3. Divide the RPK values by the “per million” scaling factor. This gives you TPM.

So you see, when calculating TPM, the only difference is that you normalize for gene length first, and then normalize for sequencing depth second. However, the effects of this difference are quite profound.

When you use TPM, the sum of all TPMs in each sample are the same. This makes it easier to compare the proportion of reads that mapped to a gene in each sample. In contrast, with RPKM and FPKM, the sum of the normalized reads in each sample may be different, and this makes it harder to compare samples directly

EXAMPLES :

* Here’s an example. If the TPM for gene A in Sample 1 is 3.33 and the TPM in sample B is 3.33, then I know that the exact same proportion of total reads mapped to gene A in both samples. This is because the sum of the TPMs in both samples always add up to the same number (so the denominator required to calculate the proportions is the same, regardless of what sample you are looking at.)
* With RPKM or FPKM, the sum of normalized reads in each sample can be different. Thus, if the RPKM for gene A in Sample 1 is 3.33 and the RPKM in Sample 2 is 3.33, I would not know if the same proportion of reads in Sample 1 mapped to gene A as in Sample 2. This is because the denominator required to calculate the proportion could be different for the two samples.

# LINKS

* **RNA seq data: Differential expression analysis**

R · [Fibrosis SMOC2 Raw Counts](https://www.kaggle.com/code/vsevolodcherepanov/rna-seq-data-differential-expression-analysis/data)

<https://www.kaggle.com/code/vsevolodcherepanov/rna-seq-data-differential-expression-analysis/notebook>

# Differential Gene Expression Analysis

<https://www.kaggle.com/code/garimabansal/differential-gene-expression-analysis/notebook>

# EDA scanpy (bioinformatics standard analysis)

<https://www.kaggle.com/code/yyoshiaki/eda-scanpy-bioinformatics-standard-analysis>

* **RNA-seq workflow: gene-level exploratory analysis and differential expression**

<https://bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html>

**RNA-seq workflow: gene-level exploratory analysis and differential expression**

<https://bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html>

Updated workflow version : <https://f1000research.com/articles/4-1070>

Here we walk through an end-to-end gene-level RNA-seq differential expression workflow using Bioconductor packages. We will start from the FASTQ files, show how these were quantified to the reference transcripts, and prepare gene-level count datasets for downstream analysis. We will perform exploratory data analysis (EDA) for quality assessment and to explore the relationship between samples, perform differential gene expression analysis, and visually explore the results.

**Contents**

* [1 Introduction](https://bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html#introduction)
  + [1.1 Experimental data](https://bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html#experimental-data)
* [2 Preparing quantification input to DESeq2](https://bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html#preparing-quantification-input-to-deseq2)
  + [2.1 Transcript quantification and *tximport* / *tximeta*](https://bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html#transcript-quantification-and-tximport-tximeta)
  + [2.2 Quantifying with *Salmon*](https://bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html#quantifying-with-salmon)
  + [2.3 Reading in data with *tximeta*](https://bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html#reading-in-data-with-tximeta)
  + [2.4 *DESeq2* import functions](https://bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html#deseq2-import-functions)
  + [2.5 SummarizedExperiment](https://bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html#summarizedexperiment)
  + [2.6 Branching point](https://bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html#branching-point)
* [3 The *DESeqDataSet* object, sample information and the design formula](https://bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html#the-deseqdataset-object-sample-information-and-the-design-formula)
  + [3.1 Starting from *SummarizedExperiment*](https://bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html#starting-from-summarizedexperiment)
  + [3.2 Starting from count matrices](https://bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html#starting-from-count-matrices)
* [4 Exploratory analysis and visualization](https://bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html#exploratory-analysis-and-visualization)
  + [4.1 Pre-filtering the dataset](https://bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html#pre-filtering-the-dataset)
  + [4.2 The variance stabilizing transformation and the rlog](https://bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html#the-variance-stabilizing-transformation-and-the-rlog)
  + [4.3 Sample distances](https://bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html#sample-distances)
  + [4.4 PCA plot](https://bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html#pca-plot)
  + [4.5 PCA plot using Generalized PCA](https://bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html#pca-plot-using-generalized-pca)
  + [4.6 MDS plot](https://bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html#mds-plot)
* [5 Differential expression analysis](https://bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html#differential-expression-analysis)
  + [5.1 Running the differential expression pipeline](https://bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html#running-the-differential-expression-pipeline)
  + [5.2 Building the results table](https://bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html#building-the-results-table)
  + [5.3 Other comparisons](https://bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html#other-comparisons)
  + [5.4 Multiple testing](https://bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html#multiple-testing)
* [6 Plotting results](https://bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html#plotting-results)
  + [6.1 Counts plot](https://bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html#counts-plot)
  + [6.2 MA-plot](https://bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html#ma-plot)
  + [6.3 Gene clustering](https://bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html#gene-clustering)
  + [6.4 Independent filtering](https://bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html#independent-filtering)
  + [6.5 Independent Hypothesis Weighting](https://bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html#independent-hypothesis-weighting)
* [7 Annotating and exporting results](https://bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html#annotating-and-exporting-results)
  + [7.1 Exporting results](https://bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html#exporting-results)
  + [7.2 Plotting fold changes in genomic space](https://bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html#plotting-fold-changes-in-genomic-space)
* [8 Removing hidden batch effects](https://bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html#removing-hidden-batch-effects)
  + [8.1 Using SVA with DESeq2](https://bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html#using-sva-with-deseq2)
  + [8.2 Using RUV with DESeq2](https://bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html#using-ruv-with-deseq2)
* [9 Time course experiments](https://bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html#time-course-experiments)
* [10 Session information](https://bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html#session-information)
* [References](https://bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html#references)

**R version**: R version 4.2.0 RC (2022-04-19 r82224)

**Bioconductor version**: 3.15

**Package**: 1.20.0

# 1 Introduction

Bioconductor has many packages which support analysis of high-throughput sequence data, including RNA sequencing (RNA-seq). The packages which we will use in this workflow include core packages maintained by the Bioconductor core team for working with gene annotations (gene and transcript locations in the genome, as well as gene ID lookup). We will also use contributed packages for statistical analysis and visualization of sequencing data. Through scheduled releases every 6 months, the Bioconductor project ensures that all the packages within a release will work together in harmony (hence the “conductor” metaphor). The packages used in this workflow are loaded with thelibrary function and can be installed by following the [Bioconductor package installation instructions](http://bioconductor.org/install/#install-bioconductor-packages).

* A published version of this workflow, including reviewer reports and comments is available at [F1000Research](http://f1000research.com/articles/4-1070). The version you are reading now differs from this one, primarily in that we now give code for performing fast **transcript quantification** followed by import in R/Bioconductor to perform gene-level analysis.
* Another Bioconductor workflow covering **differential transcript usage** (DTU) is the[rnaseqDTU](https://bioconductor.org/packages/rnaseqDTU) workflow, with the published version likewise available at [F1000Research](https://f1000research.com/articles/7-952/v3).
* If you have questions about this workflow or any Bioconductor software, please post these to the [Bioconductor support site](https://support.bioconductor.org/). If the questions concern a specific package, you can tag the post with the name of the package, or for general questions about the workflow, tag the post with rnaseqgene. Note the [posting guide](http://www.bioconductor.org/help/support/posting-guide/) for crafting an optimal question for the support site.

## **1.1Experimental data**

The data used in this workflow is stored in the [*airway*](https://bioconductor.org/packages/3.15/airway) package that summarizes an RNA-seq experiment wherein airway smooth muscle cells were treated with dexamethasone, a synthetic glucocorticoid steroid with anti-inflammatory effects (Himes et al. 2014). Glucocorticoids are used, for example, by people with asthma to reduce inflammation of the airways. In the experiment, four primary human airway smooth muscle cell lines were treated with 1 micromolar dexamethasone for 18 hours. For each of the four cell lines, we have a treated and an untreated sample. For more description of the experiment see the [PubMed entry 24926665](http://www.ncbi.nlm.nih.gov/pubmed/24926665) and for raw data see the [GEO entry GSE52778](http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE52778).

# 2 Preparing quantification input to DESeq2

As input, the count-based statistical methods, such as [*DESeq2*](https://bioconductor.org/packages/3.15/DESeq2) (Love, Huber, and Anders 2014),[*edgeR*](https://bioconductor.org/packages/3.15/edgeR) (Robinson, McCarthy, and Smyth 2009), [*limma*](https://bioconductor.org/packages/3.15/limma) with the voom method (Law et al. 2014), [*DSS*](https://bioconductor.org/packages/3.15/DSS)(Wu, Wang, and Wu 2013), [*EBSeq*](https://bioconductor.org/packages/3.15/EBSeq) (Leng et al. 2013) and [*baySeq*](https://bioconductor.org/packages/3.15/baySeq) (Hardcastle and Kelly 2010), expect input data as obtained, e.g., from RNA-seq or another high-throughput sequencing experiment, in the form of a matrix of un-normalized counts. The value in the i-th row and the j-th column of the matrix tells how many reads (or fragments, for paired-end RNA-seq) can be assigned to gene i in sample j. Analogously, for other types of assays, the rows of the matrix might correspond e.g., to binding regions (with ChIP-Seq), or peptide sequences (with quantitative mass spectrometry).

The values in the matrix should be counts or estimated counts of sequencing reads/fragments. This is important for DESeq2’s statistical model to hold, as only counts allow assessing the measurement precision correctly. It is important to **never provide counts that were pre-normalized for sequencing depth/library size**, as the statistical model is most powerful when applied to un-normalized counts, and is designed to account for library size differences internally.

## **2.1 Transcript quantification and tximport / tximeta**

A previous version of this workflow (including the published version) demonstrated how to align reads to the genome and then count the number of reads that are consistent with gene models. We now recommend a faster, alternative pipeline to genome alignment and read counting. This workflow will demonstrate how to import transcript-level quantification data, aggregating to the gene-level with tximport or tximeta. Transcript quantification methods such as [Salmon](https://combine-lab.github.io/salmon/) (Patro et al. 2017), [kallisto](https://pachterlab.github.io/kallisto/)(Bray et al. 2016), or [RSEM](http://deweylab.github.io/RSEM/) (Li and Dewey 2011) perform mapping or alignment of reads to reference transcripts, outputting estimated counts per transcript as well as effective transcript lengths which summarize bias effects. After running one of these tools, the [*tximport*](https://bioconductor.org/packages/3.15/tximport) (Soneson, Love, and Robinson 2015) or [*tximeta*](https://bioconductor.org/packages/3.15/tximeta) (Love et al. 2020) packages can be used to assemble estimated count and offset matrices for use with Bioconductor differential gene expression packages, as will be demonstrated below.

A tutorial on how to use the **Salmon software** for quantifying transcript abundance can be found [here](https://combine-lab.github.io/salmon/getting_started/). We recommend using the --gcBias [flag](http://salmon.readthedocs.io/en/latest/salmon.html#gcbias) which estimates a correction factor for systematic biases commonly present in RNA-seq data (Love, Hogenesch, and Irizarry 2016; Patro et al. 2017), unless you are certain that your data do not contain such bias.

The advantages of using the **transcript abundance quantifiers** in conjunction with **tximport/tximeta** to produce gene-level count matrices and normalizing offsets, are:

* 1. this approach corrects for any potential changes in gene length across samples (e.g. from differential isoform usage) (Trapnell et al. 2013)
  2. some of these methods are substantially faster and require less memory and disk usage compared to alignment-based methods; and
  3. it is possible to avoid discarding those fragments that can align to multiple genes with homologous sequence (Robert and Watson 2015). Note that transcript abundance quantifiers skip the generation of large files which store read alignments, instead producing smaller files which store estimated abundances, counts, and effective lengths per transcript. For more details, see the manuscript describing this approach (Soneson, Love, and Robinson 2015), and the [*tximport*](https://bioconductor.org/packages/3.15/tximport) package vignette for software details.

[*tximeta*](https://bioconductor.org/packages/3.15/tximeta) (Love et al. 2020) extends tximport, offering the same functionality, plus the additional benefit of automatic addition of annotation metadata for commonly used transcriptomes (GENCODE, Ensembl, RefSeq for human and mouse). See the [tximeta vignette](https://bioconductor.org/packages/release/bioc/vignettes/tximeta/inst/doc/tximeta.htmlm) package vignette for more details. tximeta produces a SummarizedExperiment that can be loaded easily into DESeq2 using the DESeqDataSet function, which will be demonstrated below. We will also discuss the various possible inputs into DESeq2, whether using tximport, tximeta, htseq (Anders, Pyl, and Huber 2015), or a pre-computed count matrix.

## **2.2 Quantifying with Salmon**

As mentioned above, a short tutorial on how to use Salmon can be found [here](https://combine-lab.github.io/salmon/getting_started/), so instead we will provide the code that was used to quantify the files used in this workflow. Salmon can be conveniently run on a cluster using the [Snakemake](https://snakemake.readthedocs.io/en/stable/) workflow management system (Köster and Rahmann 2012).

The following Snakemake file was used to quantify the eight samples that were downloaded from the SRA (the SRR identifier is the run identifier, and there was only one run per sample for these eight samples).