1 RNA-Seq Expression Analysis

1.1 Introduction

RNA sequencing (RNA-Seq) is a high-throughput method used to profile the transcriptome, quantify gene expression and discover novel RNA molecules. This tutorial uses RNA sequencing of from two ***Mycobacterium tuberculosis*** isolates (each with 3 replicates - 6 total samples) to walk you through transcriptome pseudo-alignment and quantification and how to profile transcriptomic differences between experimental conditions (***Sensitive*** vs. ***Resistant***) by identifying differentially expressed genes.

For an introduction to RNA-Seq principles and best practices see:

> **A survey of best practices for RNA-Seq data analysis**
>
> Ana Conesa, Pedro Madrigal, Sonia Tarazona, David Gomez-Cabrero, Alejandra Cervera, Andrew McPherson, Michał Wojciech Szcześniak, Daniel J. Gaffney, Laura L. Elo, Xuegong Zhang and Ali Mortazavi Genome Biol. 2016 Jan 26;17:13 doi:10.1186/s13059-016-0881-8

1.2 Learning Outcomes

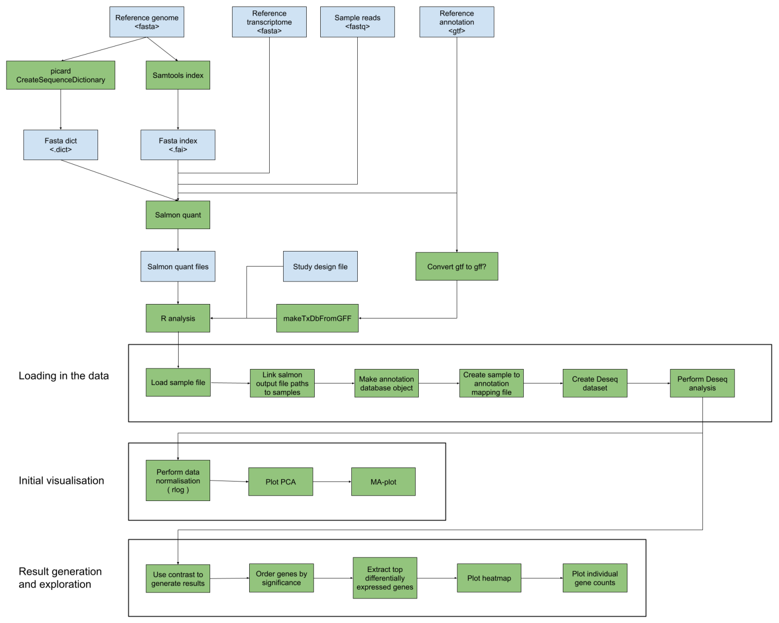
By the end of this practical, you should be familiar with:

- Using salmon to quantify transcript abundance
- Knowing how to create a study design file
- Loading the required files into R
- Creating a DESeq2 data set
- Running DESeq
- Visualising the relationship between samples
- Creating a results object
- Ranking genes by significance
- Visualisation of differential expression results

Focus on:

- Looking at what is in the files you are given, and the files you create
- How the information from different files relates and comes together
- Testing different inputs to commands to see what happens

1.3 Practical Outline


Reference genome <fasta>
Reference transcriptome <fasta>
Sample reads <fastq>
Reference annotation <gtf>
picard CreateSequenceDictionary
Samtools index
Fasta dict <.dict>
Fasta index <.fai>
Salmon quant
Salmon quant files
Study design file
Convert gtf to gff?
R analysis
makeTxDbFromGFF
Loading in the data
Load sample file
Link salmon output file paths to samples
Make annotation database object
Create sample to annotation mapping file
Create Deseq dataset
Perform Deseq analysis
Initial visualisation
Perform data normalisation (rlog)
Plot PCA
MA-plot
Result generation and exploration
Use contrast to generate results
Order genes by significance
Extract top differentially expressed genes
Plot heatmap
Plot individual gene counts

Workflow Overview. A schematic representation of each step in this practical exercise.

The main steps in all differential expression analyses are:

- Quantification - How many reads come from each gene? (***Salmon***)
- Normalisation - Dealing with biases in the data (***DESeq2***)
- Differential expression analysis (***DESeq2***)
- Visualisation and reporting (**R libraries**)

Additional Resources

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

1.4 Authors

This tutorial was developed by Jon Ambler, Phelelani Mpangase and Nyasha Chambwe, based in part, from materials from Victoria Offord and Adam Reid.

1.5 Prerequisites

This tutorial assumes that you have the following software or packages and their dependencies installed on your computer. The software or packages used in this tutorial may be updated from time to time so, we have also given you the version which was used when writing the tutorial. Where necessary, instructions for how to download additional analysis tools are given in the relevant section.

Package	Link for download/installation instructions	Version
Trimmmomatic	https://github.com/usadellab/Trimmomatic	0.39

Package	Link for download/installation instructions	Version
Salmon	https://salmon.readthedocs.io/en/latest/index.html	1.4.0
R	https://www.r-project.org/	4.0.2
tximport	https://bioconductor.org/packages/release/bioc/html/tximport.html	1.20.0
GenomicFeatures	https://bioconductor.org/packages/release/bioc/html/GenomicFeatures.html	1.44.0
DESeq2	https://bioconductor.org/packages/release/bioc/html/DESeq2.html	1.32
pheatmap	https://cran.r-project.org/web/packages/pheatmap/index.html	1.0.12

1.6 Setup

1.6.1 Create Practical Directory

Navigate to the module folder on the Virtual Machine (VM) using the following command:

[ ]:
```
cd /home/manager/course_data/rna_seq_pathogen
```

Create a working directory for the practical

[ ]:
```
mkdir practical
```

Create a copy of the practial dataset to maintain the data integrity of the original dataset.

[ ]:
```
cp /home/manager/course_data/rna_seq_pathogen/data/bacterial/* /home/manager/
↪course_data/rna_seq_pathogen/practical
```

Copy and paste the command above as a single line at your command line window.

Note: if you make a mistake at any point during this tutorial, you can reset by deleting the practical folder and restarting from this section.

Move to the practical directory

[ ]:
```
cd practical
```

1.6.2 Download Supplemental Datasets

Internet Access Required

Reference Transcriptome File

Download the GFF formatted version of the annotation file.

Unfortunately due to the lack of standardization in bioinformatics, we need both the GTF file and the GFF file for analysis (See the differences and similarities between these two file format specifications here). Salmon uses the GTF file and the GenomicFeatures library in **R** needs a GFF file. Both files contain the same annotation information, they are just formatted differently. No tools are able to reliably convert one to the other because even GFF files are not formatted consistently.

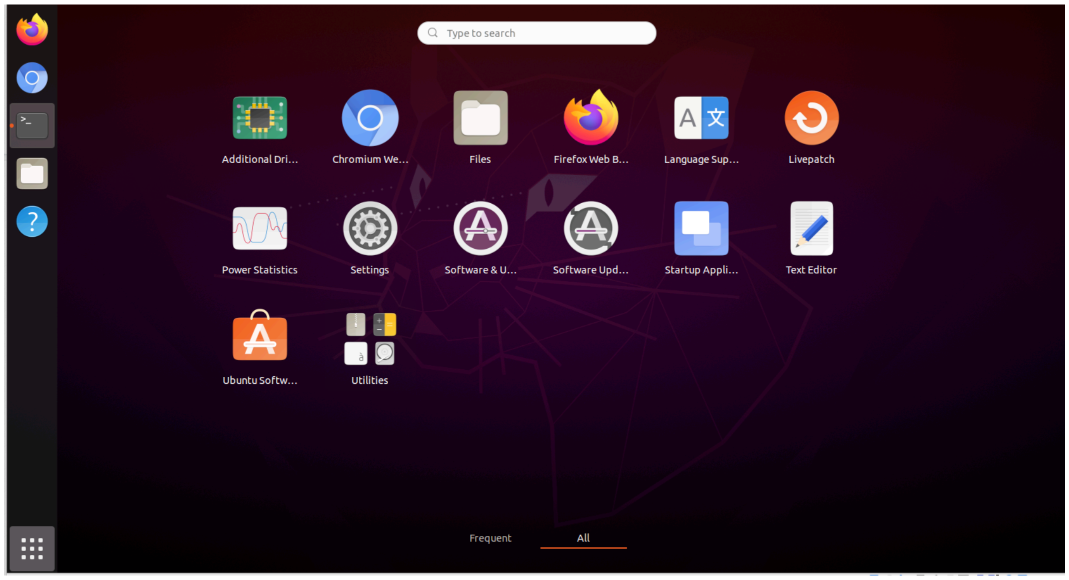
[ ]:
```
wget https://www.dropbox.com/s/4yjgbmy3dyhfoad/GCA_000195955.
↪2_ASM19595v2_genomic.gff?dl=1 -O /home/manager/course_data/rna_seq_pathogen/
↪practical/GCA_000195955.2_ASM19595v2_genomic.gff
```

Study Design File Download the study design file

The study design file is a tab separated file with 4 columns that encodes the phenotype and repeat information on the samples we will analyze in this practical. You will import this file into R to setup the differential expression analysis in the **R** Programming language. It is critical to keep track of the details of which samples are which etc. for downstream analysis purposes.

[ ]:
```
wget https://www.dropbox.com/s/6y3z9btz3bg20pn/practical_study_design.txt?dl=1␣
↪-O /home/manager/course_data/rna_seq_pathogen/practical/
↪practical_study_design.txt
```

Note: In this practical, we will use the existing study design file provided. However, in your own work, you will have to create this file on your own. You can create this file in several ways some of which could include exporting an Excel spreadsheet as a 'Tab delimited' text file. You can also use your favorite text editor.such as using the ***Text Editor*** app under the Applications menu on the virtual machine.



RData Download "DE_data.RData" for later use.

These files will be used for differential expression analysis in a later section of this tutorial.

[ ]:
```
wget https://www.dropbox.com/s/6onagsnpp9wrkrv/DE_data.RData.zip?dl=1 -O /home/
↪manager/course_data/rna_seq_pathogen/practical/DE_data.RData.zip
```

Uncompress the object:

[ ]:
```
unzip DE_data.RData.zip
```

1.6.3 Setup R Analysis Environment

Internet Access Required

In this section you will download additional R Libraries, packages of analysis tools to support differential expression analysis in **R**.

Start R by typing the following command:

[ ]:
```
R
```

The Bioconductor Project provides tools written in R for the 'analysis and comprehension of high-throughput genomic data'. Here we will install two additional software packages (GenomicFeatures and tximport) for our practical today (See additional guidelines for Bioconductor package installation).

[ ]:
```
# Bioconductor Packages
install.packages("BiocManager")
BiocManager::install("GenomicFeatures", force = TRUE)
BiocManager::install("tximport")

# CRAN Packages
install.packages("pheatmap")
```

When asked whether to:

Update all/some/none? [a/s/n]

Select no (n)

Exit R.

[ ]:
```
q()
```

2 Introducing The Tutorial Dataset

Working through this tutorial, you will investigate the differences in expression between two ***Mycobacterium tuberculosis*** isolates with different phenotypes (***Sensitive*** vs. ***Resistant***). You will analyze **6** samples across the two conditions. The experiment is setup in the following way:

run	unique_id	phenotype	repeat
N2	sample1	Resistant	1
N6	sample2	Resistant	2
N10	sample3	Resistant	3
N14	sample4	Sensitive	1

run	unique_id	phenotype	repeat
N18	sample5	Sensitive	2
N22	sample6	Sensitive	3

> **Research Question:** what genes are differentially expressed between these two isolates that can explain the differences in their phenotypes?

Check that you can see the FASTQ files in the practical directory.

[ ]:
```
ls N*.fq.gz
```

The FASTQ files contain the raw sequence reads for each sample. There will typically be four lines per read:

1. Header
2. Sequence
3. Separator (usually a '+')
4. Encoded quality value

Take a look at one of the FASTQ files.

[ ]:
```
zless N2_sub_R1.fq.gz | head
```

You can find out more about the FASTQ format at https://en.wikipedia.org/wiki/FASTQ_format.

2.1 Exercise 1

2.2 Questions

Q1: Why is there more than one FASTQ file per sample?

Hint: think about why there is a N2_sub_R1.fq.gz and a N2_sub_2.fq.gz

Q2: How many reads were generated for the N2 sample?

Hint: we want the total number of reads from both files (N2_sub_R1.fq.gz and N2_sub_2.fq.gz) so perhaps think about the FASTQ format and the number of lines for each read or whether there's anything you can use in the FASTQ header to search and count.

Q3: The three *Resistant* samples N2, N6 and N10 represent technical replicates. True or False? Comment on your answer

3 Estimate Transcript Abundance With Salmon

In this section, you will use Salmon, a transcript quantification method to estimate the number of reads in each sample that map to reference transcripts. This count is an estimate of the abundance or expression level of these transcripts in our experiment. As Salmon is an alignment free method, read quantification does not require an input BAM file. Salmon using a selective mapping algorithm to align the reads directly to a set of target transcripts such as those from a reference database for your organism.

Inputs include: * A set of target transcripts - FASTA format Reference Transcriptome `GCA_000195955.2_ASM19595v2_genomic.transcripts.fa` * Sample reads - FASTA/FASTQ files for your sample `N2_sub_R1.fq.gz.....`

See more details in:

> **Reference:**
>
> Patro R, Duggal G, Love MI, Irizarry RA, Kingsford C. **Salmon provides fast and bias-aware quantification of transcript expression.** Nat Methods. 2017 Apr;14(4):417-419. doi: 10.1038/nmeth.4197. PMID: 28263959; PMCID: PMC5600148.

3.1 Create Transcriptome Index

First, we create the necessary index files for any alignment tools downstream. The salmon index is created from the transcripts file, and while it is named ***transcripts_index*** here, you can name it anything.

[ ]:
```
salmon index -t GCA_000195955.2_ASM19595v2_genomic.transcripts.fa \
    -i transcripts_index -k 31
```

Take a look at the various files created in the **transcripts_index** folder created by salmon.

3.2 Transcript Quantification

Next we perform quantification with salmon using the transcript index folder we just created.

However, it is important to perform key quality control analysis of your sample reads (`*.fq.gz`) files before proceeding with quantification.

3.2.1 Read Quality Control Analysis

Here we demonstrate how to trim and filter the reads using `Trimmomatic`. Here we clean the reads by looking at various quality metrics including:

- **low quality reads** - filter based on a min leading base quality of 3, and trailing base quality of 3. The scan with a sliding window that cuts when the average quality is lower than 15.
- **Illumina adapter sequences** - clip out these technical artifacts using `TruSeq3-PE.fa`
- **read length** - discard any ready with a length lower than 36

[ ]:
```
trimmomatic PE N2_sub_R1.fq.gz N2_sub_R2.fq.gz \
 N2_trimmed_forward_paired.fq.gz N2_trimmed_forward_unpaired.fq.gz \
 N2_trimmed_reverse_paired.fq.gz N2_trimmed_reverse_unpaired.fq.gz \
 ILLUMINACLIP:/home/manager/miniconda/pkgs/trimmomatic-0.39-1/share/
↪trimmomatic-0.39-1/adapters/TruSeq3-PE.fa:2:30:10:2:keepBothReads \
 LEADING:3 TRAILING:3 MINLEN:36
```

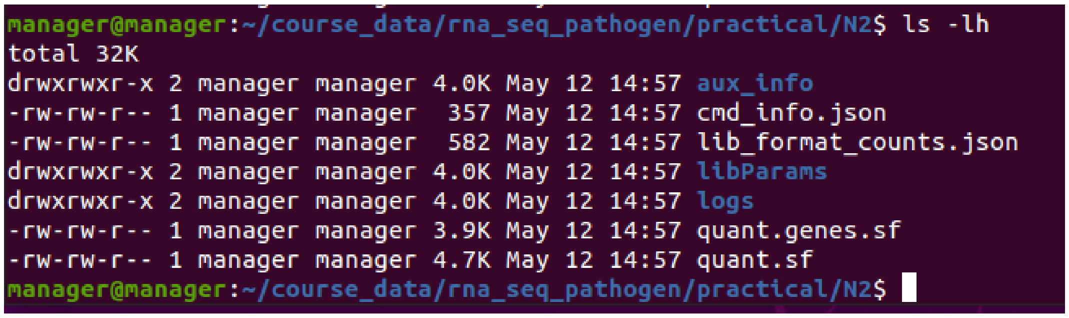
In a typically workflow, you would do fastQC analysis before and after to check the effects of the trimming, but as that is not the purpose of this section, we will skip that for now.

3.2.2 Transcript Quantificaiton using salmon

Now we used the trimmed and paired reads to estimate transcript abundance:

```
[ ]: salmon quant \
      --geneMap GCA_000195955.2_ASM19595v2_genomic.gtf \
      --threads 2 -l A \
      -i transcripts_index GCA_000195955.2_ASM19595v2_genomic.fa \
      -1 N2_trimmed_forward_paired.fq.gz \
      -2 N2_trimmed_reverse_paired.fq.gz -o N2
```

This creates a new folder named "N2" in the directory, which contains a number of files:



```
manager@manager:~/course_data/rna_seq_pathogen/practical/N2$ ls -lh
total 32K
drwxrwxr-x 2 manager manager 4.0K May 12 14:57 aux_info
-rw-rw-r-- 1 manager manager  357 May 12 14:57 cmd_info.json
-rw-rw-r-- 1 manager manager  582 May 12 14:57 lib_format_counts.json
drwxrwxr-x 2 manager manager 4.0K May 12 14:57 libParams
drwxrwxr-x 2 manager manager 4.0K May 12 14:57 logs
-rw-rw-r-- 1 manager manager 3.9K May 12 14:57 quant.genes.sf
-rw-rw-r-- 1 manager manager 4.7K May 12 14:57 quant.sf
manager@manager:~/course_data/rna_seq_pathogen/practical/N2$
```

The ***"quant"*** files are the files containing the read counts for the genes. These are what downstream tools like DESeq2 or edgeR would use for the differential expression analysis.

It is not always practical to run commands for each sample one at a time. So we can write small scripts to process multiple samples using the same set of commands as is shown here:

```
[ ]: for r1 in *_R1.fq.gz
     do
       echo $r1
       sample=$(basename $r1)
       sample=${sample%_sub_R1.fq.gz}
       echo "Processing sample: "$sample

     trimmomatic PE ${sample}_sub_R1.fq.gz ${sample}_sub_R2.fq.gz \
      ${sample}_trimmed_forward_paired.fq.gz ${sample}_trimmed_forward_unpaired.fq.
      ↪gz \
      ${sample}_trimmed_reverse_paired.fq.gz ${sample}_trimmed_reverse_unpaired.fq.
      ↪gz \
      ILLUMINACLIP:/home/manager/miniconda/pkgs/trimmomatic-0.39-1/share/
      ↪trimmomatic-0.39-1/adapters/TruSeq3-PE.fa:2:30:10:2:keepBothReads \
      LEADING:3 TRAILING:3 MINLEN:36

     salmon quant --geneMap GCA_000195955.2_ASM19595v2_genomic.gtf \
      --threads 2 -l A -i transcripts_index GCA_000195955.2_ASM19595v2_genomic.fa \
      -1 ${sample}_trimmed_forward_paired.fq.gz \
      -2 ${sample}_trimmed_reverse_paired.fq.gz \
      -o ${sample}

     done
```

3.3 Questions

In the N2 folder, take a look at the **quant.sf** file and answer the following questions:

- Q1: What does TPM stand for?
- Q2: How many reads in total are mapped to the tRNA genes? (They start with "rna-")
- Q3: Do you think this level of tRNA is acceptable?

Take a moment to discuss what the effect of large amounts of reads from tRNA and rRNA could have on normalisation.

Hint: For further help understanding the **quant.sf*** output files look at this salmon documentation page.*

4 Differential Expression Analysis with DESeq2

In this section, you will use the R Package DESeq2 to perform differential expression analysis using the salmon quantification files you generated in the previous section. For more information about DESeq can look at the manuscript describing the method:

> **Reference:**
>
> Love MI, Huber W, Anders S (2014). **"Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2."** Genome Biology, 15, 550. doi: 10.1186/s13059-014-0550-8.

We will now move to working in the **R** Programming environment: ## Load Data into the R Environment

Start **R**

[ ]:
```
R
```

Load the required libraries

R has a base set of classes and methods and tools. The additional packages we installed are a set of functions and tools designed to handle specific biological data types and support analyses and visualization such as of RNA-seq data. Here, we load those software packages that are relevant to our analysis.

[ ]:
```
library("tximportData")
library("tximport")
library("GenomicFeatures")
library("DESeq2")
library("pheatmap")
```

Load the study design table

[ ]:
```
design_file <-"practical_study_design.txt"

samples <- read.table(design_file, header=TRUE)

# Set the row names as the samples names
rownames(samples) <- samples$run

samples
```

If we look at the samples object, we can see the data from our study design file.

Link Salmon Output File Paths to Samples

Create a files object pointing to the related **quant.sf** file for each sample

[ ]:
```
root_dir <- "/home/manager/course_data/rna_seq_pathogen/practical"
files <- file.path(root_dir, samples$run, "quant.sf")
files
```

In the **files** object, we should now see the path to the **quant.sf** file in each sample's folder. This is much simpler and less error-prone than typing out each file path manually.

Use the **run** column to map samples to their file paths

[ ]:
```
names(files) <- samples$run
```

The **files** object now has the sample name linked to the file path and can be used as a way to map the information.

4.0.1 Make Annotation Database Object

Load the annotation information

[ ]:
```
path_to_gff <- "GCA_000195955.2_ASM19595v2_genomic.gff"
txdb <-makeTxDbFromGFF(path_to_gff, organism="Mycobacterium tuberculosis")
```

The **makeTxDbFromGFF** function is part of the GenomicFeatures library and is used to create a Transcript Database (txdb) object from a GFF annotation file. The TxDb class is a container for storing transcript annotations.

Extract the GENEID key values from the tbdx object

[ ]:
```
k <- keys(txdb, keytype = "GENEID")
head(k)
```

The **k** object is a list of all the gene names based on the **GENEID** as extracted from the annotation file.

Create a mapping table based on the GENEID and TXNAME info

```
[ ]: tx2gene <- select(txdb, keys = k, keytype = "GENEID", columns = "TXNAME")
     head(tx2gene)
```

Take a look at the output, you will see 2 columns. This will be used to map the gene names from the salmon files to the annotation files.

Rename the genes in the GENEID column

We do this to match the gene names found in the salmon **quant.sf** files with those found in the transcriptome and annotation files.

```
[ ]: tx2gene[["GENEID"]] <- with(tx2gene, ifelse(!grepl("rna-", TXNAME),␣
      ↪paste0("gene-", TXNAME), TXNAME))
     head(tx2gene[["GENEID"]])
```

The above command goes through the ***tx2gene*** object, and looks for where the gene name in the **TXNAME** column **DOES NOT** have the prefix **"rna-"**. In those rows, it adds the **"gene-"** prefix to the gene name in the **GENEID** column.

When the annotation file is imported by **makeTxDbFromGFF** , it removes the “gene-” prefix that is present in the quant files. This is because of the way that the annotation and transcript files are formatted.

You will see in the GFF and transcript files, that the gene ID is formatted like this:

ID=gene-Rv0005

Whereas in the GTF file it is formatted like this:

gene_id "Rv0005";

Though salmon uses the GTF file, it adds on the gene- prefix where it is missing while **makeTxDbFromGFF** removes it where it is present.

```
[ ]: txi <- tximport(files, type="salmon", tx2gene=tx2gene)
```

4.0.2 Create the DESeq data set object

```
[ ]: ddsTxi <- DESeqDataSetFromTximport(txi, colData = samples, design = ~ phenotype)
```

The inputs to create a DESeq object are:

- **txi** - the summary of transcript level abundance estimates
- **colData** - information from the study design file loaded earlier
- **design** - formula that expresses the relationship between the gene counts and the variables in the study design. Here, we are using phenotype.

The design formula is a statement that specifies how we want to model the variation in gene expression from the abundance estimates for each sample (counts). This statement is loosely saying that we expect that the level of expression of a gene is dependent on the phenotype of the bacterial isolate. If we have more than a single experimental factor to consider, we would change how we specify the design formula.

Note: For a more extensive treatment of how to setup the design formula for more complex experimental designs, read through A guide to creating design matrices for gene expression experiments.

4.0.3 Perform DESeq Analysis

Normalisation and model fitting with DESeq2

Next we used DESeq to normalize the data and fit a model that relates the gene count information to the phenotype.

[ ]:
```
ddsTxi <- DESeq(ddsTxi)
```

This step does the actual normalisation and model fitting for the data, and results in a deseq data set.

Load provided .RData for differential expression analysis:

[ ]:
```
load("DE_data.RData")
```

4.1 Exploratory Visualization

Transform data for visualization:

The **rlog** transformation provides functionality for visualization and clustering.

[ ]:
```
# R log norm
rldTxi <- rlog(ddsTxi, blind = FALSE)
```

Plot principal component analysis (PCA):

The PCA plot allow us to assess the similarities and differences between the study samples in order to determine if the data fit our expectation compared to the experimental design. This is a good quality control check to use to ensure that we did not introduce any errors during sample processing, sequencing and primary data analysis steps.

[ ]:
```
# Plot PCA
plotPCA(rldTxi, intgroup = c("phenotype"))
```

4.2 Result Generation and Exploration

Get summary of differentially expression results:

The `results` function of DESeq2 is used to get the the results of the DESeq function ranked by metrics to use to determine which genes are significantly differentially expressed.

[ ]:
```
summary(results(ddsTxi))
```

Apply LFC threshold and adjusted p-value cutoffs to get significantly differentially expressed genes:

A significance threshold (FDR, or alpha) and log2FC threshold can be passed to the `results` function to filter non-signficant differentially expressed genes.

```
[ ]: resTxi <- results(ddsTxi, alpha=0.05, lfcThreshold=0.5,␣
      ↪altHypothesis="greaterAbs")
     summary(resTxi)
```

Get significantly differentially expressed genes:

```
[ ]: sigTxi <- resTxi[!is.na(resTxi$padj) & resTxi$padj < 0.05 &␣
      ↪abs(resTxi$log2FoldChange) >= 0.5,]
     sigTxi
```

Visualize your differentially expressed genes:

Extract the rlog transformed expression values for the significant gene to use for the heatmap.

```
[ ]: matrixTxi <- assay(rldTxi)[rownames(sigTxi), ]
     matrixTxi
```

Plot heatmap:

```
[ ]: pheatmap(matrixTxi, cluster_rows=FALSE, show_rownames=TRUE, cluster_cols=TRUE)
```

4.3 Questions

Q1. How many genes are up- and down-regulated between the resistant and sensitive isolates of the Mycobacterium tuberculosis?

Hint:use the summary function on the DESeq results object.

Q2: How many genes are significantly differentially expressed (i.e., meet the LFC threshold and adjusted p-value cutoffs) between the resistant and sensitive isolates of the Mycobacterium tuberculosis? Name these genes.

Q3. What are the p-values for the significantly differentially expressed genes?

5 Key Aspects of Differential Expression Analysis

5.1 Replicates and power

’**Biological replicates** are parallel measurements of biologically distinct samples that capture random biological variation, which may itself be a subject of study or a noise source.

Technical replicates are repeated measurements of the same sample that represent independent measures of the random noise associated with protocols or equipment’

Blainey, Paul et al. “Points of significance: replication.” Nature methods vol. 11,9 (2014): 879-80. doi:10.1038/nmeth.3091

In order to accurately ascertain which genes are differentially expressed and by how much it is necessary to use replicated data. As with all biological experiments doing it once is simply not enough. There is no simple way to decide how many replicates to do, it is usually a compromise between statistical power and cost. By determining how much variability there is in the sample preparation and sequencing reactions, we can better assess how highly genes are really expressed and more accurately determine any differences. The key to this is performing biological rather than technical replicates. This means, for instance, growing up three batches of parasites, treating them all identically, extracting RNA from each and sequencing the three samples separately. Technical replicates, whereby the same sample is sequenced three times do not account for the variability that really exists in biological systems or the experimental error between batches of parasites and RNA extractions.

Note: more replicates will help improve power for genes that are already detected at high levels, while deeper sequencing will improve power to detect differential expression for genes which are expressed at low levels.

5.2 p-values vs. q-values

When asking whether a gene is differentially expressed we use statistical tests to assign a p-value. If a gene has a p-value of 0.05, we say that there is only a 5% chance that it is not really differentially expressed. However, if we are asking this question for every gene in the genome, then we would expect to see p-values less than 0.05 for many genes even though they are not really differentially expressed. Due to this statistical problem, we must correct the p-values so that we are not tricked into accepting a large number of erroneous results. **Q-values** are p-values which have been corrected for what is known as multiple hypothesis testing. Therefore, it is a qvalue of less than 0.05 that we should be looking for when asking whether a gene is signficantly differentially expressed.

5.3 Altermate software

If you have a good quality genome and genome annotation such as for model organisms e.g. human, mouse, Plasmodium etc. map to the transcriptome to determine transcript abundance. This is even more relevant if you have variant transcripts per gene as you need a tool which will do its best to determine which transcript is really expressed. Kallisto (Bray et al. 2016; PMID: 27043002 and eXpress (Roberts & Pachter, 2012; PMID: 23160280 are examples of these tools.

5.4 What do I do with a list of differentially expressed genes?

Differential expression analysis results are a list of genes which show differences between conditions. It can be daunting trying to determine what the results mean. On one hand, you may find that there are no real differences in your experiment. Is this due to biological reality or noisy data? On the other hand, you may find several thousands of genes are differentially expressed.

What can you say about that?

Other than looking for genes you expect to be different or unchanged, one of the first things to do is look at common themes in gene functionality across your list. For example, you can carry out Gene Ontology (GO) term enrichment analysis. There are many different algorithms for this, but you could annotate your genes with functional terms from GO using for instance Blast2GO (Conesa et al., 2005; PMID: 16081474) and then use TopGO (Alexa et al., 2005; PMID: 16606683) to determine whether any particular sorts of genes occur more than expected in your differentially expressed genes.

6 Normalisation

6.1 Introduction

In a previous section, we looked at estimating transcript abundance with Salmon. The abundances are reported as **transcripts per million (TPM)**, but what does TPM mean and how is it calculated?

The objectives of this part of the tutorial are:

- *understand why RNA-Seq normalisation metrics are used*
- *understand the difference between RPKM, FPKM and TPM*
- *understand how RPKM and TPM are calculated for a gene of interest*

There are many useful websites, publications and blog posts which go into much more detail about RNA-Seq normalisation methods. Here are just a couple (in no particular order):

- What the FPKM? A review of RNA-Seq expression units
- RPKM, FPKM and TPM, clearly explained
- A survey of best practices for RNA-seq data analysis
- The RNA-seq abundance zoo

6.2 Why do we use normalisation units instead of raw counts?

Raw reads counts are the number of reads originating from each transcript which can be affected by several factors:

- **sequencing depth (total number of reads)**
  The more we sequence a sample, the more reads we expect to be assigned.
- **gene/transcript length**
  The longer the gene or transcript, the more reads we expect to be assigned to it.

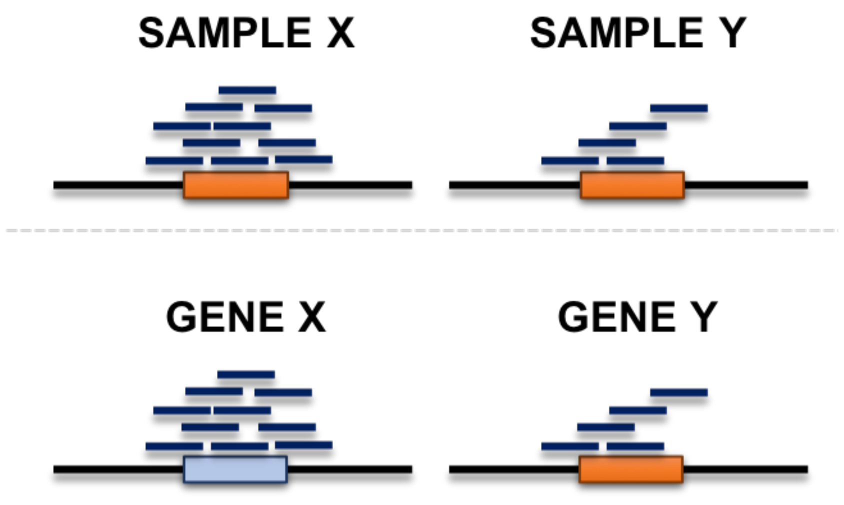


Figure 4. Effect of sequencing depth and gene length on raw read counts

Look at the top part of Figure 4. In which sample, X or Y, is the gene more highly expressed?

Neither, it's the same in both. What we didn't tell you was that the total number of reads generated for sample A was twice the number than for sample B. That meant almost twice the number of reads are assigned to the same gene in sample A than in sample B.

Look at the bottom part of Figure 4. Which gene, X or Y, has the greatest gene level expression?

Neither, they are both expressed at the same level. This time we didn't tell you that gene X is twice the length of gene Y. This meant that almost twice the number reads were assigned to gene X than gene Y.

In the top part of *Figure 4*, the gene in sample X has twice the number of reads assigned to it than the same gene in sample Y. What isn't shown is that sample X had twice the number or total reads than sample Y so we would expect more reads to be assigned in sample X. Thus, the gene is expressed at roughly the same level in both samples. In the bottom part of *Figure 4*, gene X has twice the number of reads assigned to it than gene Y. However, gene X is twice the length of gene Y and so we expect more reads to be assigned to gene X. Again, the expression level is roughly the same.

6.2.1 Reads per kilobase per million (RPKM)

Reads per kilobase (of exon) per million (reads mapped) or **RPKM** is a **within sample** normalisation method which takes into account sequencing depth and length biases.

To calculate RPKM, you first normalise by sequencing depth and then by gene/transcript length.

1. **Get your *per million* scaling factor**
   Count up the total number of reads which have been assigned (mapped) in the sample. Divide this number by 1,000,000 (1 million) to get your *per million* scaling factor (N).

2. **Normalise for sequencing depth**
   Divide the number of reads which have been assigned to the gene or transcript (C) by the *per million* scaling factor you calculated in step 1. This will give you your reads per million (RPM).

3. **Get your *per kilobase* scaling factor**
   Divide the total length of the exons in your transcript or gene in base pairs by 1,000 (1 thousand) to get your *per kilobase* scaling factor (L).
4. **Normalise for length**
   Divide your RPM value from step 2 by your *per kilobase* scaling factor (length of the gene/-transcript in kilobases) from step 3. This will give you your reads per kilobase per million or RPKM.

This can be simplified into the following equation:

$$RPKM = \frac{C}{LN}$$

Where:

- **C** is number of reads mapped to the transcript or gene
- **L** is the total exon length of the transcript or gene in kilobases
- **N** is the total number of reads mapped in millions

6.2.2 Fragments per kilobase per million (FPKM)

Fragments per kilobase per million or **FPKM** is essentially the same as RPKM except that:

- RPKM is designed for **single-end** RNA-Seq experiments
- FPKM is designed for **paired-end** RNA-Seq experiments

In a paired-end RNA-Seq experiment, two reads may be assigned to a single fragment (in any orientation). Also, in some cases, only one of those reads will be assigned to a fragment (singleton). The only difference between RPKM and FPKM is that FPKM takes into consideration that two reads may be assigned to the same fragment.

6.2.3 Transcripts per million (TPM)

Calculating the **transcripts per million** or **TPM** is a similar process to RPKM and FPKM. The main difference is that you will first normalise for length bias and then for sequencing depth bias. In a nut shell, we are swapping the order of normalisations.

1. **Get your *per kilobase* scaling factor**
   Divide the total length of the exons in your transcript in base pairs by 1,000 (1 thousand) to get your *per kilobase* scaling factor.
2. **Normalise for length**
   Divide the number of reads which have been assigned to the transcript by the per kilobase scaling factor you calculated in step 1. This will give you your reads per kilobase (RPK).
3. **Get the sum of all RPK values in your sample**
   Calculate the RPK value for all of the transcripts in your sample. Add all of these together to get your total RPK value.

4. **Get your *per million* scaling factor**
   Divide your total RPK value from step 3 by 1,000,000 (1 million) to get your *per million* scaling factor.

5. **Normalise for sequencing depth**
   Divide your RPK value calculated in step 2 by the *per million* scaling factor from step 4. You now have your transcripts per millions value or TPM.

6.3 Calculating RPKM and TPM values

To try and answer this, let's look at a worked example. Here, we have three genes (A-C) and three biological replicates (1-3).

Gene	Length	Replicate 1	Replicate 2	Replicate 3
A	2,000 bases	10	12	30
B	4,000 bases	20	25	60
C	1,000 bases	5	8	15

There are two things to notice in our dataset:

- Gene B has twice number reads mapped than gene A, possibly as it's twice the length
- Replicate 3 has more reads mapped than any of the other replicates, regardless of which gene we look at

6.3.1 Calculating RPKM

Step 1: get your per million scaling factor In the table below is the total number of reads which mapped for each of the replicates. To get our *per million* scaling factor, we divide each of these values by 1,000,000 (1 million).

Gene	Replicate 1	Replicate 2	Replicate 3
Total reads mapped	3,500,000	4,500,000	10,600,000
Per million reads	3.5	4.5	10.6

Step 2: normalise for sequencing depth We now divide our read counts by the *per million* scaling factor to get our reads per million (RPM).

Before:

Gene	Replicate 1	Replicate 2	Replicate 3
A	10	12	30
B	20	25	60
C	5	8	15

After:

Gene	Replicate 1 RPM	Replicate 2 RPM	Replicate 3 RPM
A	2.857	2.667	2.830
B	5.714	5.556	5.660
C	1.429	1.778	1.415

Step 3: get your *per kilobase* scaling factor Here we have our gene length in base pairs. For our *per kilobase* scaling factor we need to get our gene length in kilobases by dividing it by 1,000.

Gene	Length (base pairs)	Length (kilobases)
A	2,000	2
B	4,000	4
C	1,000	1

Step 4: normalise for length Finally, we divide our RPM values from step 2 by our *per kilobase* scaling factor from step 3 to get our reads per kilobase per million (RPKM).

Before:

Gene	Replicate 1 RPM	Replicate 2 RPM	Replicate 3 RPM
A	2.857	2.667	2.830
B	5.714	5.556	5.660
C	1.429	1.778	1.415

After:

Gene	Replicate 1 RPKM	Replicate 2 RPKM	Replicate 3 RPKM
A	1.43	1.33	1.42
B	1.43	1.39	1.42
C	1.43	1.78	1.42

Notice that even though replicate 3 had more reads assigned than the other samples and a greater sequencing depth, its RPKM is quite similar. And, that although gene B had twice the number of reads assigned than gene A, its RPKM is the same. This is because we have normalised by both length and sequencing depth.

6.3.2 Calculating TPM

Now we're going to calculate the TPM values for the same example data. As a reminder, here are our three genes (A-C) and three biological replicates (1-3).

Gene	Length	Replicate 1	Replicate 2	Replicate 3
A	2,000 bases	10	12	30
B	4,000 bases	20	25	60
C	1,000 bases	5	8	15

Step 1: get your *per kilobase* scaling factor Again, our gene lengths are in base pairs. For our *per kilobase* scaling factor we need to get our gene length in kilobases by dividing it by 1,000.

Gene	Length (base pairs)	Length (kilobases)
A	2,000	2
B	4,000	4
C	1,000	1

Step 2: normalise for length Now we divide the number of reads which have been assigned to each gene by the per kilobase scaling factor we just calculated. This will give us our reads per kilobase (RPK).

Before:

Gene	Replicate 1	Replicate 2	Replicate 3
A	10	12	30
B	20	25	60

Gene	Replicate 1	Replicate 2	Replicate 3
C	5	8	15

After:

Gene	Replicate 1	Replicate 2	Replicate 3
A	5	6	15
B	5	6.25	15
C	5	8	15

Step 3: get the sum of all RPK values in your sample Next, we sum the RPK values for each of our replices. This will give use our total RPK value for each replicate. To make this example scalable, we assume there are other genes so the total RPK is made up.

Gene	Replicate 1	Replicate 2	Replicate 3
A	5	6	15
B	5	6.25	15
C	5	8	15
...	...	...	...
Total RPK	**150,000**	**202,500**	**450,000**

Step 4: get your *per million* scaling factor Here, instead of dividing our total mapped reads by 1,000,000 (1 million) to get our *per million* scaling factor, we divide our total RPK values by 1,000,000 (1 million).

Gene	Replicate 1	Replicate 2	Replicate 3
Total RPK	150,000	202,500	450,000
Per million RPK	0.1500	0.2025	0.4500

Step 5: normalise for sequencing depth Finally, we divide our individual RPK values from step 2 by the *per million* scaling factor in step 4 to give us our TPM values.

Before:

Gene	Replicate 1	Replicate 2	Replicate 3
A	5	6	15
B	5	6.25	15
C	5	8	15

After:

Gene	Replicate 1	Replicate 2	Replicate 3
A	33.33	29.63	33.33
B	33.33	30.86	33.33
C	33.33	39.51	33.33

6.4 Which normalisation unit should I use?

Well, there's a lot of debate around this, so let's look at our total normalised values for each replicate.

6.4.1 RPKM

Gene	Replicate 1 RPKM	Replicate 2 RPKM	Replicate 3 RPKM
A	1.43	1.33	1.42
B	1.43	1.39	1.42
C	1.43	1.78	1.42
Total RPKM	**4.29**	**4.50**	**4.25**

6.4.2 TPM

Gene	Replicate 1	Replicate 2	Replicate 3
A	33.33	29.63	33.33
B	33.33	30.86	33.33
C	33.33	39.51	33.33
Total TPM	**100**	**100**	**100**

Notice that that total TPM value for each of the replicates is the same. This is not true for RPKM and FPKM where the total values differ. With TPM, having the same total value for each replicate makes it easier to compare the proportion of reads mapping to each gene across replicates (although you shouldn't really compare across experiments). With RPKM and FPKM, the differing total values make it much harder to compare replicates.

Congratulations, you have reached the end of this tutorial!