RNA Sequencing

Ribonucleic Acids (RNA) are a type of single-stranded nucleic acid that are present in all living cells.

To understand the RNA, it's helpful to know about the Central Dogma.

- The Central Dogma of Molecular Biology describes the transfer of information from DNA (Deoxyribonucleic Acid) to RNA to Peptide.
- First proposed in 1958 by Francis Crick, it describes how RNA molecules are synthesized from the DNA via transcription.

What is RNA Sequencing, and why do we do it?

RNA sequencing (or, RNA-Seq) examines and quantifies sequences of RNA in a sample using a type of High Throughput Sequencing called Next Generation Sequencing (NGS).

There are 3 main steps in RNA-Seq, which are:

- Preparing a sequencing library
- Sequencing
- Data Analysis

RNA Seq is basically used to measure gene expression in normal cells and then comparatively measure the gene expression in mutated cells.

Gene expression is the synthesis of a specific protein with a sequence of amino acids that is encoded within the gene – i.e. the processes of transcription and translation are collectively referred to as gene expression. It is here that high throughput sequencing tells us which genes are active and how much they are transcribed.

Workflow of an RNAseq Experiment

1. <u>Library Preparation:</u>

- Repairing is done if mRNA is degraded, else Poly-A Selection is carried out (a type of Enrichment of the Total RNA, as it selectively pulls out mRNA molecules by binding)
- Input RNA is converted to DNA (this is done to increase stability and ensure successful sequencing)
- Fragmentation is usually done to convert the DNA fragments to uniform length since different mRNA transcripts are usually of different lengths
- This is followed by Adapter Ligation

2. <u>Cluster Generation:</u>

- The flow cell is a glass slide having lanes, where each lane is composed of 2 types of oligos:

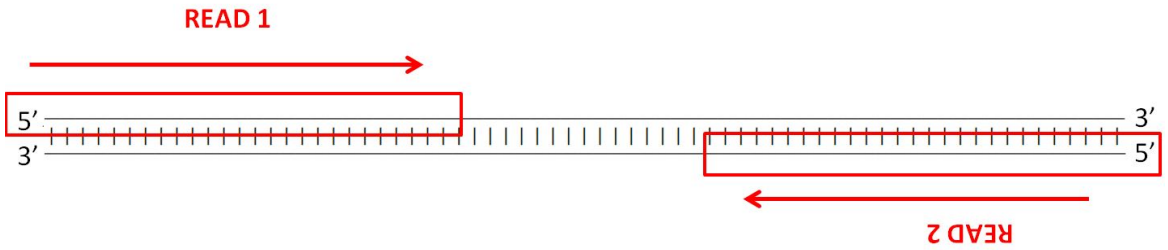
    - The first type of oligo is the one which is complementary to a transcript fragment Adapter Sequence 1. It binds to the incoming fragments and creates a reverse read strand.
    - The original fragment is washed away, and these remaining fragments are clonally amplified using bridge amplification.
    - Bridge amplification can be described as: the strands fold over, and the other Adapter Sequence hybridizes with Oligo Type 2. This oligo generates the complementary strand of the reverse read strand, thereby forming a bridge.

- This bridge is then denatured forming a forward strand (like the original one) and reverse strand. This whole process is done over and over in separate clusters and then all the reverse strands are cleaved and washed off leaving behind only the forward strands.

3. Sequencing by Synthesis (usually done by Illumina Tech)
    - The primer binds to the adapter sequence and the complementary bases are allowed to bind with the Template DNA. This is done in order to figure out the sequence of the template strand.
    - Then the fluorescent emission from the bonded nucleotides are detected by the sequencer, and the probes are subsequently washed off.
    - This process is repeated for the whole strand.

4. Analysis

- The NGS Data Output can be computationally analyzed throughout the stages of performing Quality Control on the FASTQ (FastQ files with QC info), Alignment followed by Quantification, and Normalization.
- For Paired-end Sequencing, there are usually Read 1 and Read 2 files – "*When the reads are aligned to the genome, one read should align to the forward strand, and the other should align to the reverse strand, at a higher base pair position than the first one so that they are pointed towards one another. This is known as an "FR" read – forward/reverse, in that order.*"



Forward and reverse reads in paired-end sequencing (Eric Minikel)

- Paired-end reading usually improves the ability to identify the relative positions of various reads in the genome. It also helps to resolve structural rearrangements such as gene insertions, deletions, or inversions.

What problem did we try to solve?

This study focuses on genes that are altered in both knockout samples (i.e. read 1 & 2) in a consistent direction, relative to wild-type samples.

With the aim to generate a working hypothesis, total RNAseq was conducted on collected RNA. From the information available, Ilumina Stranded Total Ligation (Ribo-Zero Plus) method was used. NextSeq 2000 sequencing run (P3 200 cycle kit, 1200M) with 2x101bp, avg 40 million reads per sample was used.

Eventually the data we got to work on was one sample of data per group – 1 WT and 1 KO, following RNA processing and clean-up. Hence, our primary investigation was to determine the levels of Profilin1 between Wild Type (WT) and Knockout (KO) samples. Additionally we also aimed to identify any variability in classically inflammatory/immune-activating pathways between the samples with IL-6 as a potential candidate.

Under the guidance of Prof. David, I learned and performed in-silico analysis of RNA Seq data.

Our Approach

Quality Control

My initial approach was to perform computational Quality Control on the data to get a better understanding of it. For this purpose, I used the tool FastQC – which is a Java-based quality control tool for high throughput sequence data.

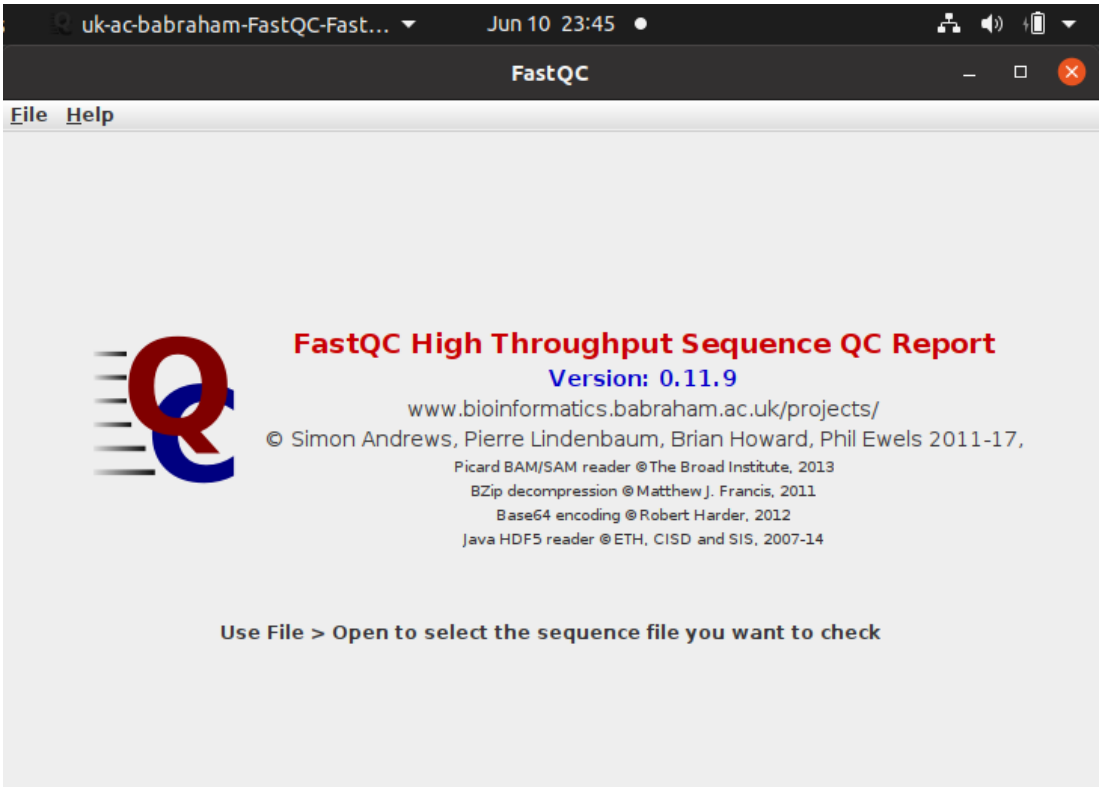
Quality Control involves verifying the library fragment lengths and concentration - basically telling us if there is anything unusual about our sequence.

As per the FastQC documentation: *"...FastQC aims to provide a simple way to do some quality control checks on raw sequence data coming from high throughput sequencing pipelines. It provides a modular set of analyses which you can use to give a quick impression of whether your data has any problems of which you should be aware before doing any further analysis."*

So after importing the data – which were gunzipped FastQ files, I installed FastQC on my Linux VM and ran it from the terminal. The installation guide of the FastQC quite helpfully explained how to run the program interactively:

- `chmod 755 fastqc:` To make the wrapper script for fastqc executable
- `./fastqc:` To run the program via the GUI

After passing the data to the program, the FastQC reports were generated. They provided a quick overview in the form of summary graphs and tables.



The main parameters where our data mainly got flagged in are:

- Per tile sequence quality: Depicts the quality scores from each tile across all of our bases to see if there was a loss in quality associated with only one part of the flow-cell. Ideally, a good plot should be blue all over.
- Per base sequence content: It plots out the proportion of each base position in a file for which each of the four normal DNA bases has been called. Ideally in a random library the lines in a plot should run parallel with each other.
- Per sequence GC content: It measures the GC content across the whole length of each sequence in a file and compares it to a modeled normal distribution of GC content.
- Sequence Length Distribution: Plots the distribution of fragment sizes in the file which was analyzed. Ideally, all sequences in a library should be of the same length.
- Sequence Duplication Levels: Counts the degree of duplication for every sequence in a library and creates a plot showing the relative number of sequences with different degrees of duplication. A high level of duplication is more likely to indicate some kind of enrichment bias (eg PCR over amplification).
- Overrepresented sequences: List of sequences which appear more than expected in the file. A sequence is considered overrepresented if it accounts for ≥ 0.1% of the total reads.

In order to analyze the reports better for the 2 reads per sample, I compiled the warnings and failures in the form of a table, which is attached below.

Summary: W - Warning, F - Fail

	Per base sequence content	Per sequence GC content	Sequence Duplication Levels	Overrepresented sequences
KO_R1	Mostly, parallel - high deviation at the very beginning (F)	Sharp peak from 27 to 45% - crosses 2500000 (F)	Unique - 25.34% (F)	GTTTCGTTTACCTTCTATAAGGCTATGATGAGCTCATGTAA TTGAAACAC (0.537%) (W)
KO_R2	Mostly, parallel - some deviation at the very beginning (F)	Sharp peak from 27 to 45% - crosses 2000000 (F)	Unique - 28.68% (F)	CGGTGGCGCACGCCTGTAGTCCCAGCTACTCGGGAGGCT GAGACAGGAGG (0.3129%) (W)
WT_R1	Mostly, parallel - high deviation at the very beginning (F)	Sharp peak from 27 to 45% - crosses 1800000 (F)	Unique - 20.73%% (F)	GTTTCGTTTACCTTCTATAAGGCTATGATGAGCTCATGTAA TTGAAACAC (0.5408%) (W)
WT_R2	Mostly, parallel - some deviation at the very beginning (F)	Sharp peak from 27 to 45% - crosses 1600000 (F)	Unique - 23.51% (F)	GGGGCGCGAAGCGGGGCTGGGCGCGCGCCGCGGCTGG ACGAGGCGCCGCC (0.2347%) (W)

FastQC also allowed us to easily export the results to an HTML based permanent report. Following which, we had a general idea about the data we were dealing with. While it would have been statistically more accurate to have a library with more samples, than a library with reads of significant depth – we proceeded to quantify the expression at the transcript level.

Quantification

The goal of quantification is to identify from which transcript each of the reads originated from and the total number of reads associated with each transcript.

For the proper analysis of the raw RNAseq data, first the garbage reads (i.e. the reads with a low Quality Score (Q.S.) or reads that are the combination of just the adapter sequences) were filtered out.

Following that we used a tool called Salmon, to quantify the reads per gene. Written mainly in C++ and CMake, Salmon's main features include being very fast and accurate, in providing transcript-level quantification estimates from RNA-seq data.

As per the Salmon documentation, *"...Salmon achieves its accuracy and speed via a number of different innovations, including the use of selective-alignment (accurate but fast-to-compute proxies for traditional read alignments), and massively-parallel stochastic collapsed variational inference."*

Also unlike pseudoalignment, Salmon's lightweight mapping procedure tracks, by default, the position and orientation of all mapped fragments.

Salmon required a set of target transcripts (here, we used the reference transcriptome of Mouse (GRCm39) - *Mus musculus*) and our RNAseq data containing the reads (i.e. the FastQ gunzipped files).

Since, we did not perform any previous alignment on the RNAseq data - we used Salmon's mapping based mode. The mapping-based mode of Salmon runs in two phases: first, indexing and then quantification.

1. Indexing the transcriptome: Indexing is a process that creates an index *to evaluate the sequences for all possible unique sequences of length k (k-mer) in the transcriptome.*

It is generally recommended that a decoy-aware transcriptome be used for indexing. The decoy sequences are regions of the target genome that are sequence similar to annotated transcripts. These are the regions of the genome most likely to cause mismapping (e.g. transcribed pseudogenes, etc.). So the main purpose of the decoy-aware index is to improve specificity and quantification accuracy – i.e. they are designed to help avoid potentially spurious mapping of reads to an annotated transcript.

We decided to run Salmon with the annotated transcriptome, using the entire genome as a decoy sequence. This was done by concatenating the genome (primary assembly file named mmus_pi.fa.gz) to the end of the transcriptome to be indexed (cdna file named mmus.fa.gz).

```
#!/bin/bash

curl
ftp.ensembl.org/pub/release-106/fasta/mus_musculus/cdna/Mus_m
usculus.GRCm39.cdna.all.fa.gz -o mmus.fa.gz

curl
ftp.ensembl.org/pub/release-106/fasta/mus_musculus/dna/Mus_mu
sculus.GRCm39.dna.primary_assembly.fa.gz -o mmus_pi.fa.gz

grep "^>" <(gunzip -c mmus_pi.fa.gz) | cut -d " " -f 1 >
decoys.txt

sed -i.bak -e 's/>//g' decoys.txt

cat mmus.fa.gz mmus_pi.fa.gz > mmus.fa.gz
```

The indexing step is independent of the reads, and only needs to be run once for a particular reference transcriptome. The index helps create a signature for each transcript in our reference transcriptome.

A parameter called "-k" was also passed (default value = 31) while indexing a transcriptome. This default value of k is optimized for reads of length 75bp or longer, i.e. the lengths from 75 bp to 105 bp are generally considered to be more suitable for evaluating gene expression.

2. Gene expression quantification: Next, the gene expression estimates were obtained from the index, using the raw fastq files directly.

The Ubuntu Terminal (v20.04.4 LTS) was used on Windows 10, to run Salmon (v1.8.0) by running the command below:

```
export PATH="/usr/local/bin/salmon-1.8.0_linux_x86_64/bin:$PATH"
salmon index -t mmus.fa.gz -d decoys.txt -p 12 -i mmus_index
--keepDuplicates
```

The directory structure consisted of:

```
/mnt/c/users/<username>/desktop/Salmon
      ↳/data
            ↳/F2_KO_S23
            ↳/F7_WT_S22
      ↳/mmus_index
      ↳qntSc.sh
```

The main folder, Salmon, consisted of 2 subdirectories called data (consisting of the reads for the samples) and mmus_index (the index we built) and the qntSc.sh file (script for running Salmon).

Then to quantify the reads per gene, the following part of the script was executed:

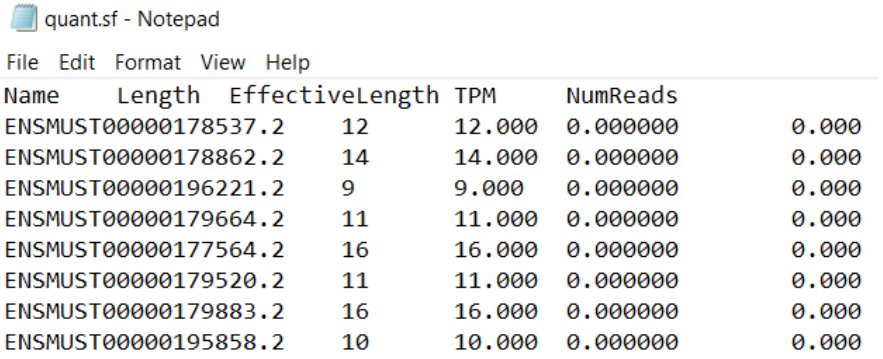
```
for fn in data/F2_KO_S23;
do
samp=`basename ${fn}`
echo "Processing sample ${samp}"
salmon quant -i mmus_index -l A \
         -1 ${fn}/${samp}_R1_001.fastq.gz \
         -2 ${fn}/${samp}_R2_001.fastq.gz \
         -p 8 --validateMappings -o quants/${samp}_quant
done


for fn in data/F7_WT_S22;
do
samp=`basename ${fn}`
echo "Processing sample ${samp}"
salmon quant -i mmus_index -l A \
         -1 ${fn}/${samp}_R1_001.fastq.gz \
```

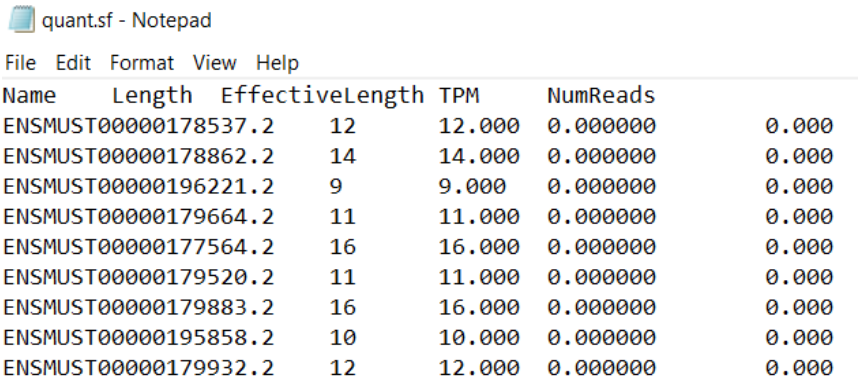
```
        -2 ${fn}/${samp}_R2_001.fastq.gz \
        -p 8 --validateMappings -o quants/${samp}_quant
done
```

3. Results: The results were stored in a directory called quants, which again had 2 separate subdirectories called F2_KO_S23_quant and F7_WT_S22_quant. Additionally, inside these 2 subdirectories there are the respective `quant.sf` files.

This file is basically the quantification file in which each row corresponds to a transcript, listed by Ensembl ID, and the columns correspond to metrics for each transcript:



F2_KO_S23 Results



F7_WT_S22 Results

The metrics included are:

- **Length** – of the transcript in base pairs (bp)
- **Effective Length** – represents the various factors that affect the length of transcript (i.e degradation, technical limitations of the sequencing platform)
- **TPM (Transcripts Per Million)** – Salmon outputs 'pseudocounts' which predict the relative abundance of different isoforms in the form of TPM. Hence, it is a commonly used normalization method and is computed based on the effective length of the transcript.
- **NumReads** – the estimated number of reads (an estimate of the number of reads drawn from this transcript given the transcript's relative abundance and length)

Differential Gene Expression Analysis

Differential expression analysis means taking the normalized read count data and performing statistical analyses, to discover quantitative changes in expression levels between experimental groups. For example, statistical testing is used to judge whether, for a given gene, an observed difference in read counts is significant, i.e. whether it is greater than what's expected just due to natural random variation.

The goal of differential expression testing and analysis is to determine which genes are expressed at different levels between conditions. These genes can offer biological insight into the processes affected by the conditions of interest.

We used a tool called DESeq2 for DGE Analysis. DESeq2 is based on negative binomial (NB) distributions & Likelihood Ratio Tests, and it helps to identify genes expressed differentially between the test and the reference groups of each pairwise contrast.

Specifics

- Requirements: R version 4.2.0
- Software used: RStudio
- Libraries used
    - GenomicFeatures (retrieves and manages transcript-related features)
    - readr
    - tximport
    - DESeq2

Steps

We constructed a DESeqDataSet from transcript abundance files and tximport. The tximport-to-DESeq2 approach uses estimated gene counts from the transcript abundance quantifiers, and not the normalized counts.

We performed transcript quantification with Salmon, and then the data was imported with tximport, which produces a list. Then we used the DESeqDataSetFromTximport() function to make a DESeq Dataset (dds) – on which we planned to calculate the vst and plot an LFC [since we're doing an one-on-one analysis]

1. Making a TxDb object from a .gtf file: Transcript information was extracted from a .gtf file by using the makeTxDbFromGFF function. GTF stands for Gene Transfer Format and has the same structure as a .gff file. The fields are:

```
<seqname> <source> <feature> <start> <end> <score> <strand>
<frame> [attributes] [comments]
```

We downloaded the gene annotation file (above) for mice and passed it to the function to build a TxDb object from it:

```
gtf_file <- "Mus_musculus.GRCm39.106.chr.gtf.gz"
txdb <- makeTxDbFromGFF(gtf_file)
```

The structure of the TxDb object can be viewed as follows:

```
> keytypes(txdb)
[1] "CDSID"    "CDSNAME"  "EXONID"   "EXONNAME" "GENEID"   "TXID"     "TXNAME"
> columns(txdb)
 [1] "CDSCHROM"   "CDSEND"     "CDSID"      "CDSNAME"    "CDSPHASE"   "CDSSTART"
 [7] "CDSSTRAND"  "EXONCHROM"  "EXONEND"    "EXONID"     "EXONNAME"   "EXONRANK"
[13] "EXONSTART"  "EXONSTRAND" "GENEID"     "TXCHROM"    "TXEND"      "TXID"
[19] "TXNAME"     "TXSTART"    "TXSTRAND"   "TXTYPE"
```

2. Mapping the keys of the TxDb object together: We queried the TxDb object with the key="TXNAME" i.e. Transcript Names, and mapped it against the respective Gene Names:

```
k <- keys(txdb, keytype="TXNAME")
tx_map <- AnnotationDbi::select(txdb, k, columns=c("TXNAME","GENEID"), "TXNAME")
```

```
> k
   [1] "ENSMUST00000193812" "ENSMUST00000082908" "ENSMUST00000192857"
   [4] "ENSMUST00000161581" "ENSMUST00000192183" "ENSMUST00000193244"
   [7] "ENSMUST00000194454" "ENSMUST00000193450" "ENSMUST00000194935"
  [10] "ENSMUST00000195361" "ENSMUST00000240255" "ENSMUST00000192738"
  [13] "ENSMUST00000193658" "ENSMUST00000134384" "ENSMUST00000027036"
```

```
> tx_map
              TXNAME             GENEID
1   ENSMUST00000193812 ENSMUSG00000102693
2   ENSMUST00000082908 ENSMUSG00000064842
3   ENSMUST00000192857 ENSMUSG00000102851
4   ENSMUST00000161581 ENSMUSG00000089699
5   ENSMUST00000192183 ENSMUSG00000103147
```

This mapping was saved to a .csv file, which is later read to a vector `tx2gene`. Additionally, the paths to the quant.sf files for the 2 samples were also read to a vector, as:

```
> quant_files
                  F2_KO_S23                   F7_WT_S22
"data//F2_KO_S23/quant.sf" "data//F7_WT_S22/quant.sf"
```

3. Using tximport to make a txi object from the transcript-gene mappings & counts: These were passed as parameters to the function `tximport(files, type = "salmon", tx2gene = NULL, ignoreTxVersion = FALSE)`. This would list the counts per gene in which sample, as:

```
> txi$counts
                  F2_KO_S23 F7_WT_S22
ENSMUSG00000000001  3298.408    71.684
ENSMUSG00000000003     0.000     0.000
ENSMUSG00000000028   114.598     1.000
ENSMUSG00000000037    28.033     0.000
ENSMUSG00000000049   391.873     1.000
```

As input, DESeq2 expects count data obtained from RNA-seq experiments, in the form of a matrix of integer values. *The value in the i-th row and the j-th column of the matrix tells how many reads can be assigned to gene i in sample j.* However, the values in the matrix should be un-normalized counts or estimated counts of sequencing fragments.

4. Making the DESeq Dataset: Finally we passed the txi object to the `DESeqDataSetFromTximport()` function after labeling the conditions, to create the dds which looks like:

```
> dds
class: DESeqDataSet
dim: 35635 2
metadata(1): version
assays(2): counts avgTxLength
rownames(35635): ENSMUSG00000000001 ENSMUSG00000000003 ...
  ENSMUSG00001074846 ENSMUSG00002076083
rowData names(0):
colnames(2): F2_KO_S23 F7_WT_S22
colData names(1): condition
```

Every DESeqDataSet object must have an associated design formula. *The design formula expresses the variables which will be used in modeling.*

This is what the counts in the DESeq Dataset look like:

```
> counts(dds)
                   F2_KO_S23 F7_WT_S22
ENSMUSG00000000001      3298        72
ENSMUSG00000000003         0         0
ENSMUSG00000000028       115         1
ENSMUSG00000000037        28         0
ENSMUSG00000000049       392         1
```

5. Calculating the varianceStabilizingTransformation (vst): VST is a form of log2 transformation of the data. It can be calculated on a dds with a design of ~1.

```
> vsd@assays@data@listData
[[1]]
                  F2_KO_S23 F7_WT_S22
ENSMUSG00000000001  9.768909  9.836164
ENSMUSG00000000003  7.865796  7.865796
ENSMUSG00000000028  8.231092  8.126688
ENSMUSG00000000037  8.054010  7.865796
ENSMUSG00000000049  8.563716  8.116161
```

6. Computing the Log Fold Change (LFC): LFC of a DESeq2 dataset is based on a negative binomial distribution – where simply subtracting the log2 values of KO from log2 values of WT results in the log2 fold change. For example, if the log2 fold change of 1.5 for a specific gene in the “WT vs KO comparison” means that the expression of that gene is increased in WT relative to KO by a multiplicative factor of 2^1.5 ≈ 2.82.

```
lfc_df$lfc <- lfc_df$F2_KO_S23 - lfc_df$F7_WT_S22
```

7. Finding the upregulated and downregulated genes: Genes are up/down regulated during development or in different pathways. This phenomenon, named gene regulation, is very important to measure as it reflects how cells react to the changes in their environments. Here, it was needed to calculate how the genes changed in the KO sample. If the tpm expression in KO was higher then the gene was upregulated, else if the tpm expression in KO was lower then the gene was downregulated.

A criteria was set to filter the tpms by >= 1, or equivalently their vsd counts by >= 8.5. After that, the top 10 upregulated and downregulated genes were filtered and stored in separate dataframes.

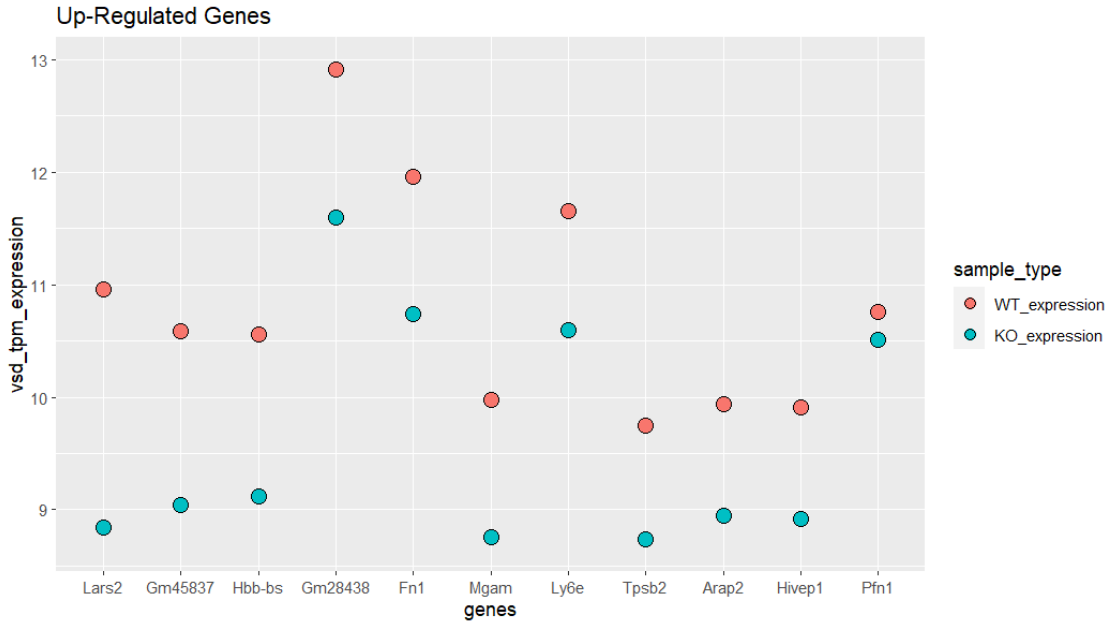
8. Extracting Profilin: As the effect of the Profilin gene was being studied, the said gene was extracted separately. Since it is downregulated, it is stored in the downregulated gene dataframe.

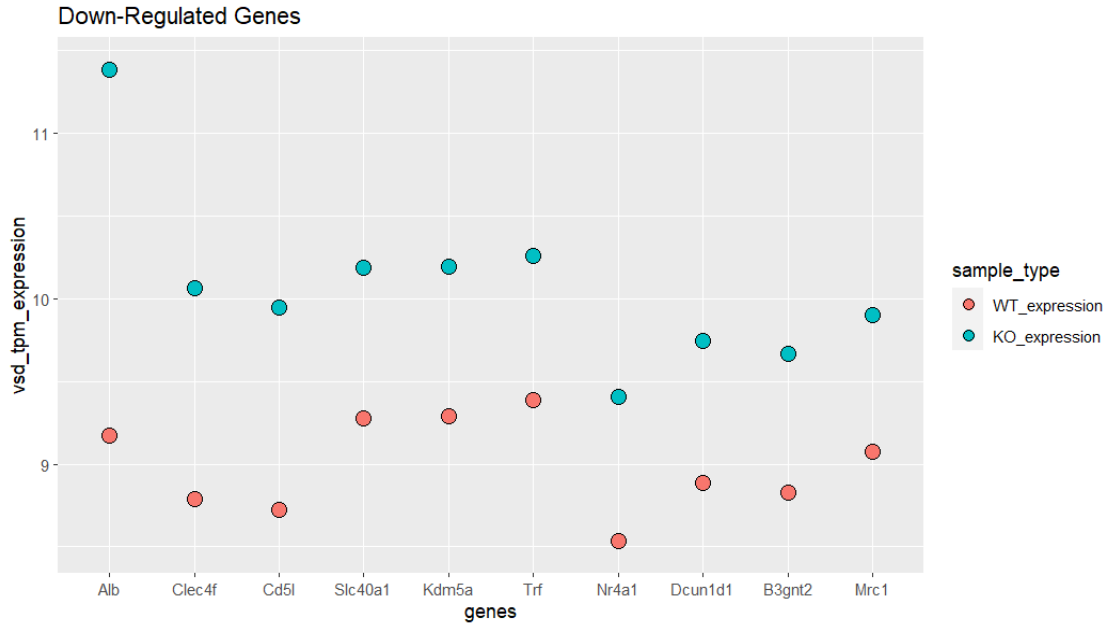
```
pfn <- og_df[og_df$gene_symbols=="Pfn1", ]
```

The expressions are as follows:

	gene_symbols	lfc	WT_expression	KO_expression
11	Pfn1	-0.2525936	10.75995	10.50736

9. Visualization: Dot plots were used to visualize the upregulated and downregulated gene vsd expression, using the ggplot2 library.





Future Scope

RNASeq is usually carried out with many replicates. This project had a workflow slightly different than the typical, as we did not possess replicates (one v/s one comparison).

In future, a more typical RNASeq experiment can be carried out with better data (i.e. with replicates).

References

- DNA, RNA and genes: https://geo.libretexts.org/Courses/University_of_California_Davis/GEL_098-16%3A_Geobiology_(Sumner)/Text/3%3A_Genes_to_Enzymes/DNA%2C_RNA%2C_Genes_and_Chromosomes
- FastQC Documentation: https://www.bioinformatics.babraham.ac.uk/projects/fastqc/
- Salmon Documentation: https://salmon.readthedocs.io/en/latest/salmon.html
- StatQuest: A gentle introduction to RNA-seq https://www.youtube.com/watch?v=tlf6wYJrwKY
- Illumina Paired-End Sequencing: https://www.youtube.com/watch?v=-8fG9ruvbe4
- Salmon: fast and bias-aware quantification of transcript expression using dual-phase inference - https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5600148/
- https://hbctraining.github.io/Intro-to-rnaseq-hpc-salmon/lessons/04_quasi_alignment_salmon.html
- Forward and reverse reads in paired-end sequencing: https://www.cureffi.org/2012/12/19/forward-and-reverse-reads-in-paired-end-sequencing/
- How does salmon deal with decoy? https://www.biostars.org/p/456231/
- What is DGE Analysis?
    - https://www.ebi.ac.uk/training/online/courses/functional-genomics-ii-common-technologies-and-data-analysis-methods/rna-sequencing/performing-a-rna-seq-experiment/data-analysis/differential-gene-expression-analysis/#:~:text=Differential%20expression%20analysis%20means%20taking,expression%20levels%20between%20experimental%20groups.
    - https://hbctraining.github.io/Training-modules/planning_successful_rnaseq/lessons/sample_level_QC.html
- DESeq2 Documentation: http://bioconductor.org/packages/devel/bioc/vignettes/DESeq2/inst/doc/DESeq2.html
- TxDb Documentation: https://www.rdocumentation.org/packages/GenomicFeatures/versions/1.24.4/topics/makeTxDbFromGFF
- Tximport Documentation: https://www.rdocumentation.org/packages/tximport/versions/1.0.3/topics/tximport
- VST Documentation: https://www.rdocumentation.org/packages/DESeq/versions/1.24.0/topics/varianceStabilizingTransformation