Group Project: Identification of NMD sensitive RNAs and isoforms with long-read direct RNA sequencing(DRS)

Tobia Ochsner, Michael Cibien, Gonzalo Cardenal Antolin (GonzaloCardenalAl)

2024-01-11

Load packages

```r
suppressPackageStartupMessages({
library(limma)
library(ggplot2)
library(DESeq2)
library(tximport)
library(tidyverse)
library(biomaRt)
library(pheatmap)
library(RColorBrewer)
library(gplots)
library(pcaExplorer)
library(IsoformSwitchAnalyzeR)
library(calibrate)
library(MASS)
library(gridExtra)
})
```

Introduction

Biological Background – NMD

Nonsense-mediated mRNA decay (NMD) is a eukaryotic pathway that is responsible for degradation of not only aberrant, but also some endogeneous mRNA, that would result in physiologically active proteins. This pathway therefore plays a pivotal role in post-transcriptional gene regulation by eliminating mRNA transcripts that could otherwise lead to the synthesis of truncated or malfunctioning proteins. The underlying mechanisms by which NMD selectively targets mRNAs for degradation are not fully elucidated, however, key proteins implicated in this process have been identified, including UPF1, an RNA helicase; SMG1, a phosphatidylinositol-kinase-related kinase; SMG6, an endonuclease; and SMG5 and SMG7, which function as adaptor proteins. (Karousis et al. 2021) (Karousis and Mühlemann 2022)

Historically, it was posited that the primary role of NMD was to target mRNAs containing premature stop codons. Recent research, however, indicates that the scope of NMD is broader, encompassing additional features of mRNAs. To explore these additional features, particularly in endogenous mRNAs, researchers have employed strategies such as the knockdown of key NMD proteins. This approach increases the intracellular concentration of NMD-targeted mRNA isoforms, facilitating their study through subsequent RNA sequencing (RNA-seq) experiments.

Gene and isoform expression analysis from direct RNA sequencing (DRS)

In conventional differential gene expression analysis short-read sequencing is widely used. However, short-read RNA sequences have limitations in identifying and quantifying gene transcript isoforms due to RNA fragmentation, bias introduction in the reverse transcription into cDNA and the subsequent amplification by PCR. DRS could act as a potential remedy to this problem, since it sequences full-length transcripts, while also characterizing RNA modifications, polyA tails and not needing prior PCR amplification.(Aird et al. 2011) (Steijger et al. 2013)

It has been shown by Josie Gleeson et al that DRS has the potential to quantify both genes and transcript isoforms in an unbiased manner allowing for a multitude of analyses: gene and isoform quantification; differential gene expression analysis; differential isoform usage analysis (DUI); discovery of novel isoforms. However, the key challenges of long-read RNA sequencing – namely sequencing depth – remains and impacts the sensitivity of isoform identification with a high probability.(Gleeson et al. 2022)

Goal of this Study

We were lucky to receive DRS data from a follow up project on their paper by Dr. Evan Karousis. They assessed endogenous targets of NMD using a combination of long-read and short-read sequencing and found that the juxtaposition of short-read sequencing with long-read sequencing enabled the identification of novel NMD-targeted mRNA isoforms. (Karousis et al. 2021)

By comparing the results of their previous work to the results of this study, where we use DRS, we aim to answer the question of whether DRS alone possesses sufficient resolution to delineate the full spectrum of NMD-targeted transcripts. Data Background The data was kindly provided by Dr. Evangelos Karousis, who generated the data while working as post-doctoral researcher in the laboratory of Prof. Dr. Oliver Mühlemann, at the University of Bern in 2019. Knockdown experiments on HeLa cells were performed by RNA interference (RNAi). For this, plasmids expressing shRNAs against SMG6 and SMG7 were transfected into the "dKD" cells, whereas in the "Scr" (control) cells plasmids targeting nothing were transfected. polyA+ mRNA was then isolated, and direct RNA-seq was performed using the Flowcell SQK-RNA002 from Oxford Nanopore. The MinKNOW instrument software generated fast5 files, on which basecalling was performed using GUPPY (VERSION?) (from Oxford Nanopore Technologies).

Preprocessing

Quality Control (QC)

Data was already classified in different folders based on the Quality Control applied with Guppy. This step was done by the lab who provided the experimental data. Two scripts were generated to confirm the filtering heuristics applied during the quality control. We confirmed that the filtering criteria was filtering out reads with q-score lower than 7. (Scripts available on the github repo under the name: "filter.r" & "checkGuppyFilter.r"). We also used Nanoplot, which provided with a report of different sequencing quality indicators (e.g. mean & median read quality, weighed histogram of read lengths…). These reports are also available in the group project GitHub repo. Also, we had reads from samples NP07 and NP08 divided in two folders NP07 & NP07B, and NP08 & NP08B. The reason for this is because during the sequencing process this stops in between so they had to restart the sequencing. Both subsamples were then merged during the mapping in one fatsq file. The results provided by Nanoplot confirmed that our samples were of optimal quality (mean read quality (q-score) > 7 and coherent read length distribution, sequencing error rates, and number of reads).

The sequencing of DRS encloses a higher difficulty. As opposed to sequencing cDNA, mRNA contains dynamic base modifications, e.g. the most abundant internal modification is N6-methyladenosine (m6A) (Roundtree et al. 2017). This increases the complexity of mRNA direct sequencing and leads to higher error rate, as when a modified nucleotide goes through the ONT MiniON pore, the signal will be incorrect. Therefore, if we want to study this modifications the quality criteria should be lower.

Mapping and quantification

To map long reads, the current gold standard is using minimap2. We ran two mappings, one to the genome and another one to the transcriptome. In the genomic mapping, we use Homo_sapiens.GRCh38.dna.primary_assembly.fa as the reference genome for each sample. In the transcriptome mapping, we use the cdna reference from the Ensmbl database: Homo_sapiens.GRCh38.cdna.all.fa .

In the quantification, we quantify the genes with Feature Counts, which provides a mode "-L" tailored for long reads. The annotation file was obtained from the Ensembl database Homo_sapiens.GRCh38.110.gtf.vWith this mode we obtain the raw counts, needed by Deseq2 for the differential expression analysis. For the transcripts and different isoforms quantification we used NanoCount. NanoCount is a recently released tool specifically developed to quantify Oxford Nanopore direct RNA sequcing (DRS) reads. (Gleeson et al. 2022)

Statistical Analysis

Transcript Coverage

We run a script that outputs a series of plots and statistics. The plots allow us to assess the coverage of the reads against the theoretical full length of their respective isoforms. For our research purpose this is specially helpful. DRS aims to sequence full length transcripts but, at the time of sequencing, there can be mRNA partly degradated mRNA or reads might not be sequence to their full length. This could lead to ambiguous alignnments where secondary alignments would hinder the quantification.

```
Rscript BamSlam.R rna NP05-sorted-transcriptomic-aln.bam NP05_BamSlam_output
Rscript BamSlam.R rna NP06-sorted-transcriptomic-aln.bam NP06_BamSlam_output
Rscript BamSlam.R rna NP07-sorted-transcriptomic-aln.bam NP07_BamSlam_output
Rscript BamSlam.R rna NP08-sorted-transcriptomic-aln.bam NP08_BamSlam_output
```

With that we can observe

(write something like this after observing results) In total, 53% of sequin reads were full-length compared to 38% of 5Y reads (Figure 1F). In comparison, using the primary alignments from minimap2 without NanoCount filtering gave a median 5Y coverage of 0.74, with 46% of sequin reads and 29% of 5Y reads being full-length. However, while a significant proportion of reads are full-length, there is considerable room for improvement through sequencing software advances and by preventing degradation during RNA extraction and library preparation.

Import Data

```
samples <- list.files("counts_files")

#Genomic data
raw_genomic <- sapply(samples, function(sample) {
  file <- paste0("counts_files/", sample, "/","primary-genomic-quant")
  quant <- read.table(file, header=TRUE, row.names=1, sep="\t")
  gene_id <- rownames(quant)
  raw <- quant[,6]
  raw_counts <- setNames(raw, gene_id)
  return(raw_counts)
})

raw_genomic <- as.data.frame(raw_genomic)

raw_genomic <- raw_genomic %>%
  rownames_to_column(var = "gene_name") %>%
  arrange(gene_name) %>%
  column_to_rownames(var = "gene_name")

head(raw_genomic)
```

```
                NP05-Scr1 NP06-dKD1 NP07-dKD2 NP08-Scr2
ENSG00000000003        62        50        83       100
ENSG00000000005         0         0         0         0
ENSG00000000419       118       111       184       182
ENSG00000000457         0         0         4         2
ENSG00000000460         0         0         0         0
ENSG00000000938         0         1         0         0
```

```
#Transcriptomic data
np05_tcrm = read.table(file = 'counts_files/NP05-Scr1/transcriptomic-quant.tsv', sep = '\t
head(np05_tcrm)
```

```
     transcript_name         raw est_count      tpm transcript_length
1  ENST00000532829.5 0.008402080  7582.558 8402.080               912
2 ENST00000222247.10 0.006247496  5638.128 6247.496               634
3 ENST00000355968.10 0.005250192  4738.099 5250.192               818
4  ENST00000504562.1 0.004519781  4078.930 4519.781               505
```

```
5  ENST00000530721.5 0.004504106  4064.785 4504.106                 796
6  ENST00000230050.4 0.004190554  3781.816 4190.554                 503
```

We are working with direct RNA Long-reads. As we can observe, NanoCounts outputs est_count and tpm. Tpm and estimated counts are not normalised by transcript length as it is usually done with Illumina data. The reason is that in DRS one read is supposed to represent a single transcript molecule starting from the polyA tail, even if the fragment doesn't extend to the 5' end.

Est_count is the estimated counts obtained by multiplying the raw abundance by the number of primary alignments, the latter is the estimated counts obtained by multiplying the raw abundance by 1M. As we observed above, the full-length transcript coverage was not higher than 30% for any of the samples. Thereby, we decided to continue the analysis with tpm.

```
np05_tcrm = data.frame(transcript_name = np05_tcrm$transcript_name, tpm_np05 = np05_tcrm$t

np06_tcrm = read.table(file = 'counts_files/NP06-dKD1/transcriptomic-quant.tsv', sep = '\t
np06_tcrm = data.frame(transcript_name = np06_tcrm$transcript_name, tpm_np06 = np06_tcrm$t

np07_tcrm = read.table(file = 'counts_files/NP07-dKD2/transcriptomic-quant.tsv', sep = '\t
np07_tcrm = data.frame(transcript_name = np07_tcrm$transcript_name, tpm_np07 = np07_tcrm$t

np08_tcrm = read.table(file = 'counts_files/NP08-Scr2/transcriptomic-quant.tsv', sep = '\t
np08_tcrm = data.frame(transcript_name = np08_tcrm$transcript_name, tpm_np08 = np08_tcrm$t

tpm_transcriptome <- Reduce(function(x, y) merge(x, y, by = "transcript_name", all = TRUE)

# Rename columns to specify the dataset source
tpm_transcriptome <- as.data.frame(tpm_transcriptome)
tpm_transcriptome  <- tpm_transcriptome  %>%
  arrange(transcript_name) %>%
  column_to_rownames(var = "transcript_name")

head(tpm_transcriptome)
```

```
                      tpm_np05    tpm_np06   tpm_np07   tpm_np08
ENST00000000233.10 309.1542913 240.5310318 283.702478 336.004874
ENST00000000412.8   76.4575129  96.7413900  93.289284  94.624125
ENST00000000442.11   7.9517822   1.4499922   2.569157   8.118034
ENST00000001008.6    0.5540399   0.0000000   0.000000   0.000000
ENST00000001146.7    0.0000000   0.5624499   0.804218   2.201090
ENST00000002125.9   11.0807987  10.1240990   7.036907   9.337958
```

Exploratory Analysis

Once we obtain the genomic and transcriptomic counts of our samples, we run a initial exploratory analysis.

```r
colSums(raw_genomic)
```

```
NP05-Scr1 NP06-dKD1 NP07-dKD2 NP08-Scr2 
   882803    877859   1239173   1323920 
```

```r
colSums(tpm_transcriptome)
```

```
tpm_np05 tpm_np06 tpm_np07 tpm_np08 
   1e+06    1e+06    1e+06    1e+06 
```

The transcript counts have the same number of total counts between samples, whereas the raw genomic counts have different number of total counts. We can normalise between the samples with scaling factors to have the same number of total counts between samples.

```r
median_all_counts = median(colSums(raw_genomic))
scaling_factors = median(colSums(raw_genomic))/colSums(raw_genomic)
normalised_genomic=sweep(raw_genomic, 2, scaling_factors, `*`)

colSums(normalised_genomic)
```

```
NP05-Scr1 NP06-dKD1 NP07-dKD2 NP08-Scr2 
  1060988   1060988   1060988   1060988 
```

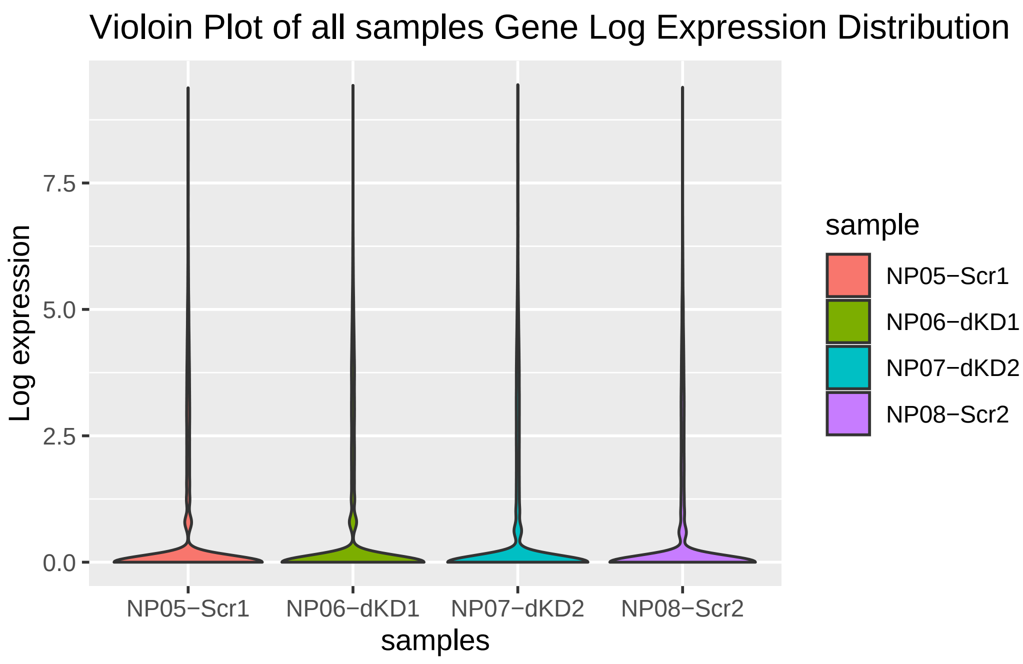
```r
log_normalised_genomic=log1p(normalised_genomic)

melted_data <- reshape2::melt(log_normalised_genomic)
```

```
No id variables; using all as measure variables
```

```r
colnames(melted_data) <- c("sample","Log_normalised_expression")

violin_plot_allgenes <- ggplot(melted_data, aes(x=sample, y=Log_normalised_expression, fil
  geom_violin() +
  labs(title='Violoin Plot of all samples Gene Log Expression Distribution', x='samples',y
  theme(plot.title = element_text(hjust = 0.5))
```

```
List of 1
 $ plot.title:List of 11
  ..$ family       : NULL
  ..$ face         : NULL
  ..$ colour       : NULL
  ..$ size         : NULL
  ..$ hjust        : num 0.5
  ..$ vjust        : NULL
  ..$ angle        : NULL
  ..$ lineheight   : NULL
  ..$ margin       : NULL
  ..$ debug        : NULL
  ..$ inherit.blank: logi FALSE
  ..- attr(*, "class")= chr [1:2] "element_text" "element"
 - attr(*, "class")= chr [1:2] "theme" "gg"
 - attr(*, "complete")= logi FALSE
 - attr(*, "validate")= logi TRUE
```

```
  print(violin_plot_allgenes)
```



All samples have similar gene expression distributions by looking at the violin plots.

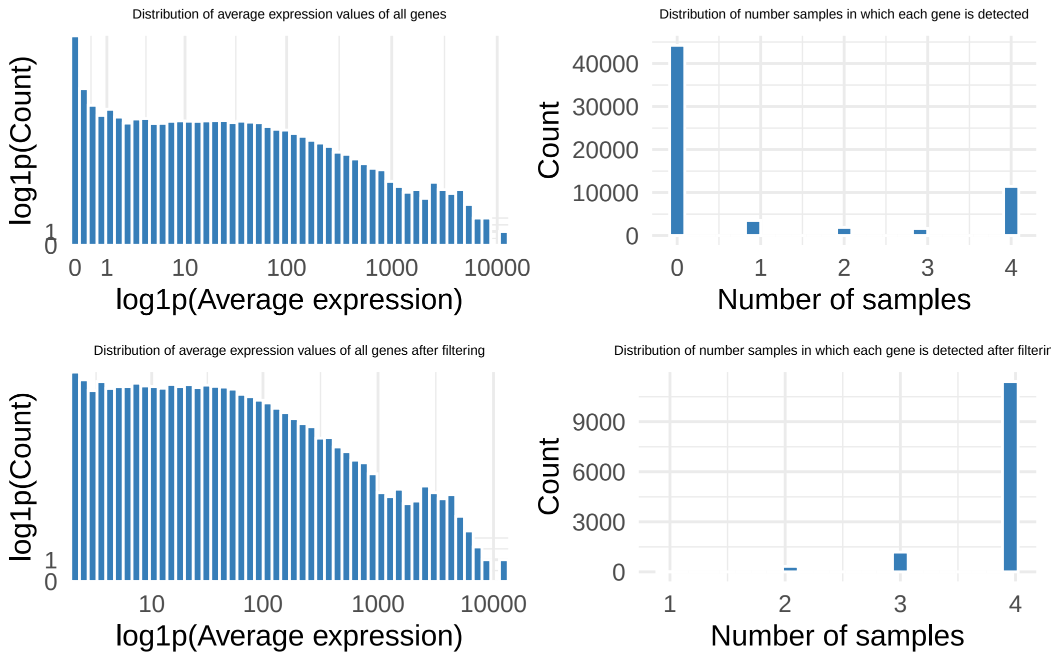
```
avg_genomic <- rowMeans(raw_genomic)

par(mfrow = c(1, 2))
layout(matrix(1:2, nrow = 1))
par(mar = c(3, 3, 2, 1))
hist(avg_genomic)
hist(log1p(avg_genomic))

layout <- matrix(c(1, 2), nrow = 1)

plot <- ggplot(data = as.data.frame(avg_genomic), mapping = aes(x = avg_genomic)) +
  geom_histogram(
    color = "white",
    fill = brewer.pal(n = 3, name = "Set1")[2],
    bins = 50
  ) +
  labs(title = "Distribution of average expression values of all genes",
       x = "Average expression",
       y = "Count") +
  theme_minimal() +
```

```r
  theme(plot.title = element_text(hjust = 0.5))

log_plot <- ggplot(data = as.data.frame(avg_genomic), mapping = aes(x = log(avg_genomic +
  geom_histogram(
    color = "white",
    fill = brewer.pal(n = 3, name = "Set1")[2],
    bins = 50
  ) +
  labs(title = "Log Distribution of average expression values of all genes",
       x = "log10(Average expression + 1)",
       y = "Count") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))

# Arrange and display the plots
grid.arrange(plot, log_plot, layout_matrix = layout)
```

```r
avg_genomic <- rowMeans(raw_genomic)
layout <- matrix(c(1, 2, 3, 4), nrow = 2, byrow = TRUE)

plot_1 <- ggplot(data = as.data.frame(avg_genomic), mapping =  aes(x = avg_genomic)) +
  geom_histogram(
    color = "white",
    fill = brewer.pal(n = 3, name = "Set1")[2],
    bins = 50
  ) +
  scale_x_continuous(
    breaks = c(0, 1, 10, 100, 1000, 10000, 20000),
    trans = "log1p",
    expand = c(0, 0)
  ) +
  scale_y_continuous(breaks = c(0, 1),
                     expand = c(0, 0),
                     trans = "log1p") +
  labs(title = "Distribution of average expression values of all genes",
       x = "log1p(Average expression)",
       y = "log1p(Count)") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5, size = 5))

num_detected_genes <- rowSums(raw_genomic > 0)
```

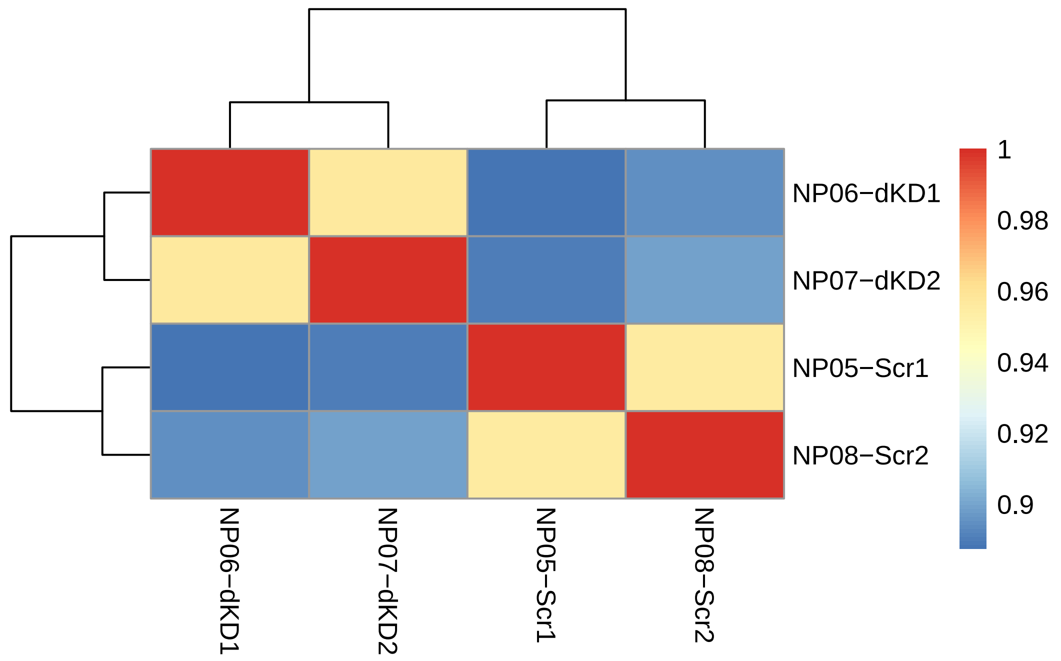
```r
plot_2 <- ggplot(data = as.data.frame(num_detected_genes), mapping =  aes(x = num_detected
  geom_histogram(
    color = "white",
    fill = brewer.pal(n = 3, name = "Set1")[2],
    bins = 23
  ) +
  labs(title = "Distribution of number samples in which each gene is detected",
       x = "Number of samples",
       y = "Count") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5, size = 5))


# filter expressed genes
# threshold: genes must be detected in at least half of the samples
#             or the average counts must be >= 1

expressed <- rowSums(raw_genomic) >= 5
num_filtered_expressed_genes <- rowSums(raw_genomic[expressed,] > 0)

avg_genomic <- data.frame(avg_genomic)
plot_3 <- ggplot(data = as.data.frame(avg_genomic[expressed,]), mapping =  aes(x = avg_gen
  geom_histogram(
    color = "white",
    fill = brewer.pal(n = 3, name = "Set1")[2],
    bins = 50
  ) +
  scale_x_continuous(
    breaks = c(0, 1, 10, 100, 1000, 10000, 20000),
    trans = "log1p",
    expand = c(0, 0)
  ) +
  scale_y_continuous(breaks = c(0, 1),
                     expand = c(0, 0),
                     trans = "log1p") +
  labs(title = "Distribution of average expression values of all genes after filtering",
       x = "log1p(Average expression)",
       y = "log1p(Count)") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5,  size = 5))
```

```
plot_4 <- ggplot(data = as.data.frame(num_filtered_expressed_genes), mapping =  aes(x = nu
  geom_histogram(
    color = "white",
    fill = brewer.pal(n = 3, name = "Set1")[2],
    bins = 23
  ) +
  labs(title = "Distribution of number samples in which each gene is detected after filter
       x = "Number of samples",
       y = "Count") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5,  size = 5))

grid.arrange(plot_1, plot_2, plot_3, plot_4, layout_matrix = layout)
```



We filter our genes to remove very low counts, removing genes with counts < 5. By looking at the plots in the right size of the figure, we can observe that our filtering criteria filtered not only all the genes not expressed in any of the samples but also most of the genes only expressed in one individual sample.

```
#Plot heatmap
corr_pearson <- cor(log1p(raw_genomic[expressed,]), method = "spearman")
pheatmap(corr_pearson)
```



The control and the knockdown samples correlate, which indicates there is a differential expression between the knockdown and the control. This again indicates the NMD knockdown might be working and causing a differential gene expression.

Differential Gene Expression Analysis

```
#DESeq input
raw_genomic<- as.matrix(raw_genomic[expressed,])
condition <- factor(c("control","knockdown","knockdown","control")) #NP05-Scr1, NP06-dKD1,
(coldata <- data.frame(row.names=colnames(raw_genomic), condition))
```

```
          condition
NP05-Scr1   control
NP06-dKD1 knockdown
NP07-dKD2 knockdown
NP08-Scr2   control
```

```
# Make DESeq dataset
dds <- DESeqDataSetFromMatrix(countData=raw_genomic, colData=coldata, design=~condition)
dds
```

```
class: DESeqDataSet
dim: 12939 4
metadata(1): version
assays(1): counts
rownames(12939): ENSG00000000003 ENSG00000000419 ... ENSG00000292366
  ENSG00000292372
rowData names(0):
colnames(4): NP05-Scr1 NP06-dKD1 NP07-dKD2 NP08-Scr2
colData names(1): condition
```

```
# Run DESeq2 pipeline
dds <- DESeq(dds)
res <- DESeq2::results(dds)
res
```

```
log2 fold change (MLE): condition knockdown vs control
Wald test p-value: condition knockdown vs control
DataFrame with 12939 rows and 6 columns
                 baseMean log2FoldChange     lfcSE       stat    pvalue
                <numeric>      <numeric> <numeric>  <numeric> <numeric>
ENSG00000000003  71.15309     -0.3257766  0.316642 -1.0288486  0.303551
ENSG00000000419 143.64052     -0.0706369  0.227744 -0.3101595  0.756440
ENSG00000000457   1.24498      0.9671047  2.550072  0.3792461  0.704505
ENSG00000001036  16.90446      0.0497611  0.664812  0.0748498  0.940334
ENSG00000001084   7.01765     -0.5919908  0.995149 -0.5948766  0.551926
...                   ...            ...       ...        ...       ...
ENSG00000292348 189.15052      0.1779944  0.197726   0.900206 0.3680104
ENSG00000292357  10.22278      1.4422616  0.848663   1.699451 0.0892342
ENSG00000292358  41.25677     -0.0652091  0.411644  -0.158411 0.8741326
ENSG00000292366  31.81156      0.7044188  0.484724   1.453238 0.1461577
ENSG00000292372   3.62745      1.2337631  1.475494   0.836170 0.4030594
                     padj
                <numeric>
ENSG00000000003  0.562073
ENSG00000000419  0.890040
ENSG00000000457        NA
ENSG00000001036  0.977413
ENSG00000001084  0.764897
...                   ...
ENSG00000292348  0.624072
ENSG00000292357  0.264707
ENSG00000292358  0.949564
```

```
ENSG00000292366  0.363055
ENSG00000292372        NA
```

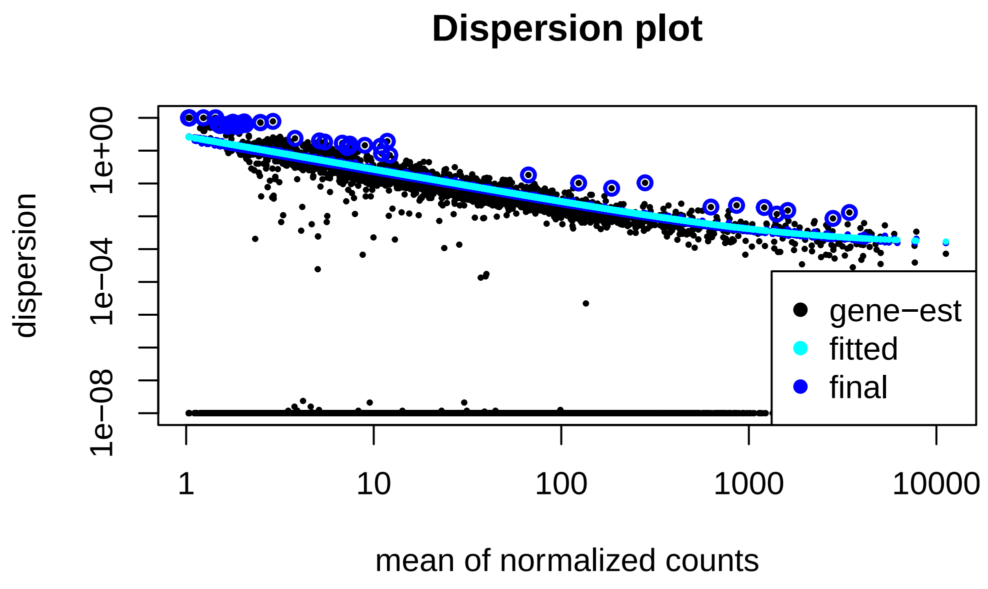
```
#DESeq2 results
table(res$padj<0.05)
```

```
FALSE  TRUE 
 7271  1654 
```

```
res <- res[order(res$padj), ]

# Merge with normalized count data
resdata <- merge(as.data.frame(res), as.data.frame(counts(dds, normalized=TRUE)), by="row.
names(resdata)[1] <- "Gene"
resdata$DE <-resdata$padj<0.05
head(resdata)
```

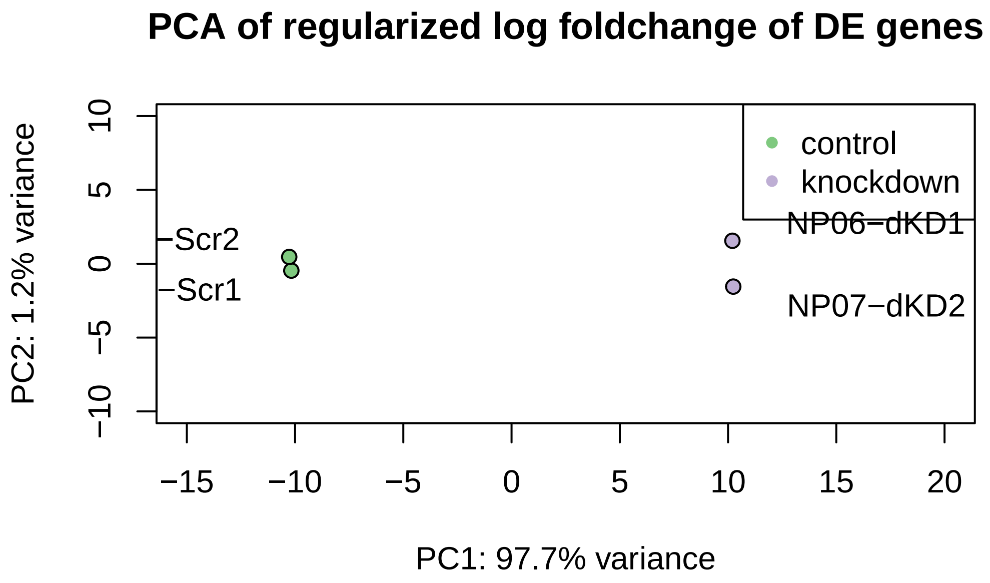
```
             Gene  baseMean log2FoldChange      lfcSE     stat        pvalue
1 ENSG00000224032 1000.8274       4.307767 0.13936391 30.91021 8.709885e-210
2 ENSG00000169242  886.1724       3.720117 0.13127462 28.33843 1.162292e-176
3 ENSG00000107262 1221.1706       2.555759 0.09930271 25.73705 4.501466e-146
4 ENSG00000210082 7819.4672       1.723809 0.06830571 25.23668 1.586237e-140
5 ENSG00000116761  605.5634       3.286235 0.14590196 22.52358 2.438256e-112
6 ENSG00000211459 3961.3526       1.490358 0.06967320 21.39070 1.630984e-101
           padj NP05-Scr1 NP06-dKD1 NP07-dKD2 NP08-Scr2   DE
1 7.773572e-206   89.5916  1837.283  1975.093  101.3427 TRUE
2 5.186729e-173  130.0919  1611.611  1681.869  121.1169 TRUE
3 1.339186e-142  351.0027  2086.889  2088.384  358.4071 TRUE
4 3.539291e-137 3424.1174 11737.182 12278.729 3837.8402 TRUE
5 4.352287e-109  101.8644  1114.679  1084.593  121.1169 TRUE
6  2.426088e-98 2055.6977  6056.650  5632.055 2101.0074 TRUE
```

```
# Plot dispersions
plotDispEsts(dds, main="Dispersion plot", genecol = "black", fitcol = "cyan", finalcol = "
```



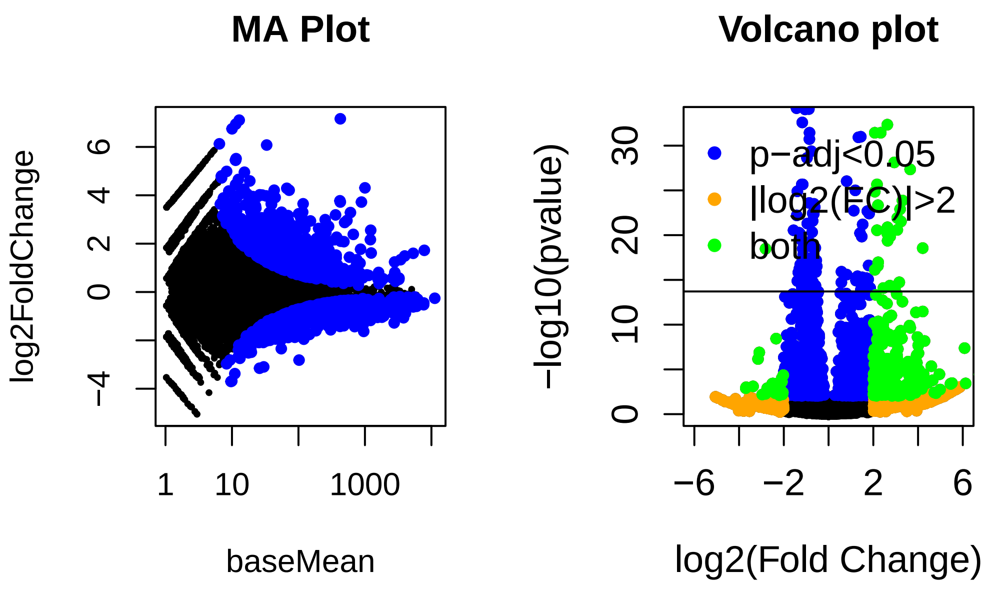
There’s a trend where dispersion decreases as the mean of normalized counts increases, which is typical in RNA-seq data due to biological variability being more pronounced in genes with low expression levels. The model (fitted line) captures the overall trend of the dispersion estimates well, as indicated by the fitted points closely following the line.

```
rld <- rlogTransformation(dds) #applies a regularized log transformation to the dds(DESeqD

#Set colours for plotting
mycols <- brewer.pal(8, "Accent")[1:length(unique(condition))]

# PCA
rld_pca <- function(rld, intgroup = "condition", ntop = 500, colors = NULL, legendpos = "t

  rv <- rowVars(assay(rld))
  select <- order(rv, decreasing = TRUE)[seq_len(min(ntop, length(rv)))]
  pca <- prcomp(t(assay(rld)[select, ]))
  fac <- factor(apply(as.data.frame(colData(rld)[, intgroup, drop = FALSE]), 1, paste, col

  pc1var <- round(summary(pca)$importance[2, 1] * 100, digits = 1)
  pc2var <- round(summary(pca)$importance[2, 2] * 100, digits = 1)
  pc1lab <- paste0("PC1: ", as.character(pc1var), "% variance")
  pc2lab <- paste0("PC2: ", as.character(pc2var), "% variance")


  plot(PC2 ~ PC1, data = as.data.frame(pca$x), bg = colors[fac], pch = 21, xlab = pc1lab,
  with(as.data.frame(pca$x), textxy(PC1, PC2, labs = rownames(pca$x), cex = textcx))
  legend(legendpos, legend = levels(fac), col = colors, pch = 20)
}

rld_pca(rld, colors = mycols, intgroup = "condition", xlim = c(-15, 20), ylim = c(-10, 10)
```



PCA further confirms there is little variability between biological replicates but high variability between gene expression of control and knockdown, indicating that the DRS long reads are capturing the variability of the NMD knockdown at a gene expression level.

```
# MA Plot
maplot <- function (res, thresh = 0.05, labelsig = FALSE, textcx = 1, ...) {
  with(res, plot(baseMean, log2FoldChange, pch = 20, cex = 0.5, log = "x", ...))
  with(subset(res, padj < thresh), points(baseMean, log2FoldChange, col = "blue", pch = 20
  if (labelsig) {
    require(calibrate)
    with(subset(res, padj < thresh), textxy(baseMean, log2FoldChange, labs = Gene, cex = t
  }
}

# Volcano Plot
volcanoplot <- function (res, lfcthresh = 2, sigthresh = 0.05, xlab = "log2(Fold Change)",
  with(res, plot(log2FoldChange, -log10(pvalue), pch = 20, xlab = xlab, cex.axis = 1.2, ce
  with(subset(res, padj < sigthresh), points(log2FoldChange, -log10(pvalue), pch = 20, col
  with(subset(res, abs(log2FoldChange) > lfcthresh), points(log2FoldChange, -log10(pvalue)
  with(subset(res, padj < sigthresh & abs(log2FoldChange) > lfcthresh), points(log2FoldCha
  legend(legendpos, xjust = 1, yjust = 1, legend = c(paste("p-adj<", sigthresh, sep = ""),
}

# Set up the layout
par(mfrow = c(1, 2), mar = c(5, 4, 4, 2) + 0.1)

# Plot MA Plot
maplot(resdata, main = "MA Plot")

# Plot Volcano Plot
volcanoplot(resdata, lfcthresh = 2, sigthresh = 0.05, xlim = c(-6, 6), ylim = c(0, 33), le
```



```
# Reset the par settings to default after plotting
par(mfrow = c(1, 1), mar = c(5, 4, 4, 2) + 0.1)
```

Differential Transcript Usage

To asses w

```
sampleID = c("tpm_np05","tpm_np06","tpm_np07","tpm_np08")
myDesign = data.frame(sampleID= sampleID, condition = condition)
tpm_transcriptome$isoform_id <- rownames(tpm_transcriptome)

aSwitchList <- importRdata(
  isoformCountMatrix   = tpm_transcriptome,
  isoformRepExpression = tpm_transcriptome,
  designMatrix         = myDesign,
  isoformExonAnnoation = "files/Homo_sapiens.GRCh38.110.chr_patch_hapl_scaff.gtf",
  isoformNtFasta       = "files/transcriptome_reference.fa",
  showProgress = FALSE
)

SwitchListFiltered <- preFilter(
  switchAnalyzeRlist = aSwitchList,
  geneExpressionCutoff = 5,
  isoformExpressionCutoff = 5,
  removeSingleIsoformGenes = TRUE)

SwitchListAnalyzed <- isoformSwitchTestDEXSeq(
  switchAnalyzeRlist = SwitchListFiltered,
  reduceToSwitchingGenes=TRUE,
  reduceFurtherToGenesWithConsequencePotential = FALSE,
  alpha = 0.05,
  dIFcutoff = 0.1,
  onlySigIsoforms = FALSE
)

extractSwitchSummary(SwitchListAnalyzed)
summary_isoforms <- SwitchListAnalyzed$isoformFeatures
write.csv(summary_isoforms[(summary_isoforms$iso_biotype == "nonsense_mediated_decay"),],
```

```
new_nmd = read.csv(file = 'NMD_exons_to_long_reads_transcriptome_mapping.csv', sep = ',',
```

Splicing Analysis

Furthermore, we use SplAdder (**kahles2016spladder?**) to perform a differential splicing analysis similar to (Karousis et al. 2021). SplAdder first generates a splicing graph based on the RNA sequencing data and extracts alternative splicing events (compared to the reference genome). The detected splicing events correspond to single or multiple skipped exons, intron retentions, alternative 3’ or 5’ splicing, and mutually exclusive exons. In a second step, it runs a differential test in order to find events that occur with a significantly different frequency in the knockdown samples compared to wildtype.

```
echo "build splicing graph"

python -m spladder.spladder build --annotation $annotationFile \
                                  --bams $wt1,$wt2,$kd1,$kd2 \
                                  --outdir $outFolder

echo "run differential splicing analyis"

python -m spladder.spladder test \
              --conditionA $wt1,$wt2 \
              --conditionB $kd1,$kd2 \
              --parallel 24 \
              --outdir $outFolder
```

```
# we load all by SplAdder detected events and merge them into one table

alt_3prime <- as.data.frame(read.table("splicing/test_results_C3_alt_3prime.tsv", sep = "\
alt_5prime <- as.data.frame(read.table("splicing/test_results_C3_alt_5prime.tsv", sep = "\
exon_skip <- as.data.frame(read.table("splicing/test_results_C3_exon_skip.tsv", sep = "\t"
intron_retention <- as.data.frame(read.table("splicing/test_results_C3_intron_retention.ts
mult_exon_skip <- as.data.frame(read.table("splicing/test_results_C3_mult_exon_skip.tsv",
mutex_exons <- as.data.frame(read.table("splicing/test_results_C3_mutex_exons.tsv", sep =


alt_3prime <- transform(alt_3prime, Type="alt_3prime")
alt_5prime <- transform(alt_5prime, Type="alt_5prime")
exon_skip <- transform(exon_skip, Type="exon_skip")
intron_retention <- transform(intron_retention, Type="intron_retention")
mult_exon_skip <- transform(mult_exon_skip, Type="mult_exon_skip")
mutex_exons <- transform(mutex_exons, Type="mutex_exons")

splicing_events <- Reduce(function(x, y) merge(x, y, all=TRUE), list(
  alt_3prime,
  alt_5prime,
  exon_skip,
  intron_retention,
  mult_exon_skip,
  mutex_exons
))
```

```
# filter only significant events
splicing_events <- splicing_events[splicing_events$p_val_adj < 0.05,]
```

We first plot the number of events per event type and the number of unique genes per event type:

```
event_counts_per_event_type <- splicing_events %>%
  group_by(Type) %>%
  summarise(UniqueCount = n_distinct(event_id))

gene_counts_per_event_type <- splicing_events %>%
  group_by(Type) %>%
  summarise(UniqueCount = n_distinct(gene_id))

ggplot(event_counts_per_event_type, aes(x = Type, y = UniqueCount)) +
  geom_bar(stat = "identity") +
  ylim(0, 250) +
  labs(x = "Splicing Event Type",
       y = "Number Events") +
  theme_minimal() +
  scale_x_discrete(guide = guide_axis(n.dodge = 2)) +
  ggplot(gene_counts_per_event_type, aes(x = Type, y = UniqueCount)) +
  ylim(0, 250) +
  geom_bar(stat = "identity") +
  labs(x = "Splicing Event Type",
       y = "Number Genes With Events") +
  scale_x_discrete(guide = guide_axis(n.dodge = 2)) +
  theme_minimal()
```

Furthermore, we can use SplAdder to confirm different splicing behavior previously reported in literature. One example is the apoptosis-modulating Bcl-2-associated athanogene-1 (BAG-1). (Karousis et al. 2021) reported that a BAG1 isoform with an included alternative exon is stabilized upon NMD inhibition. This effect can also be observed in our data set:

```
echo "run differential splicing analyis"

python -m spladder.spladder viz \
              --range gene ENSG00000107262 \
              --track coverage wildtype:$wt1,$wt2 knockdown:$kd1,$kd2 \
              --track event exon_skip \
              -O plot --format png \
```