Neutrophil RNASeq Analysis

Paul Horn

17/10/2021

1. Overview

We extracted mRNA from primary human neutrophil granulocytes obtained from four healthy volunteer donors which were treated with either vehicle control, recombinant human VAP-1 or rhVAP-1 + Benzylamine (Bz). mRNA library preparation and sequencing were performed by Eurofins using their INVIEW Transcriptome Discover pathway. Paired end 2x150 bp reads were obtained with the NovaSeq 6000 S2 PE150 XP protocol on the Genome Sequencer Illumina HiSeq platform. Trimming of adapter and primer sequences, quality trimming and filtering were performed with `Cutadapt v3.4` (http://dx.doi.org/10.14806/ej.17.1.200) and `fastp v0.20.1` (https://doi.org/10.1093/bioinformatics/bty560), and quality assessment of untrimmed and trimmed reads was performed with `FastQC v0.11.9` (Andrews, S. (2010). FastQC: A Quality Control Tool for High Throughput Sequence Data [Online]. Available online at: http://www.bioinformatics.babraham.ac.uk/projects/fastqc/), before transcript expression was quantified using the bias-aware pseudoalignment tool `salmon v1.5.2` (Patro, R., Duggal, G., Love, M. I., Irizarry, R. A., & Kingsford, C. (2017). Salmon provides fast and bias-aware quantification of transcript expression. Nature Methods.). A full decoy-away transcriptome index of the gencode v29 transcriptome (hg38) (https://www.gencodegenes.org/human/release_29.html) was generated using `salmon index`, run on the public GenePattern server (cloud.genepattern.org) and this index was used to quantify transcript pseudocounts using `salmon quant` with the following options: `-l A -p 6 --seqBias --gcBias --validateMappings --incompatPrior 0.01`. Count data were imported into R using the `tximport v1.18.0` package (doi: 10.12688/f1000research.7563.1) and differential gene expression was performed using `DESeq2 v1.30.1` (doi: 10.1186/s13059-014-0550-8), with the exception of the *vehicle vs fMLP* comparison, which was performed with `edgeR v3.32.1` (doi: 10.1093/bioinformatics/btp616) due to the lack of replicates in the fMLP control. Gene set enrichment analyses on gene ontology pathways and canonical pathway gene sets from the MSigDb dababase (v7.4) were performed with the `ClusterProfiler` package in R.

2. Quality Control

2.1. Loading

First, we load the salmon output into R, using the default tximport workflow and summarised the expression values on the gene level. Table 1 gives the sample identity including treatments.

```
dirs <- list.dirs(path = "./output_v29", recursive = F)

quant.files <-
  lapply(dirs, function(x)
    list.files(x, pattern = "quant.sf", full = T)) %>% unlist()

colData <- read_csv("colData.csv") %>%
  as.data.frame(rownames = colData$ID) %>%
```

```
  mutate(treatment = factor(
    treatment,
    levels = c("vehicle", "VAP", "VAP+BZ", "BZ", "H2O2", "fMLP")
  ))

knitr::kable(colData[, 1:3], row.names = F, caption = "Table 1: Sample identity")
```

Table 1: Table 1: Sample identity

ID	patient	treatment
c18	1	vehicle
c19	1	VAP
c20	1	VAP+BZ
c21	2	vehicle
c22	2	VAP
c23	2	VAP+BZ
c24	3	vehicle
c25	3	VAP
c26	3	VAP+BZ
c27	4	vehicle
c28	4	VAP
c29	4	VAP+BZ
c30	control	BZ
c31	control	fMLP
c32	control	H2O2

```
# txdb <- makeTxDbFromGFF(file = "http://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_human/release_29/g
# saveDb(txdb, file = "gencode.v29.annotation.sqlite")

txdb.filename <- "gencode.v29.annotation.sqlite"
txdb <- loadDb(txdb.filename)
k <- keys(txdb, keytype = "TXNAME")
tx2gene <- AnnotationDbi::select(txdb, k, "GENEID", "TXNAME")
head(tx2gene)
```

```
##              TXNAME            GENEID
## 1 ENST00000456328.2 ENSG00000223972.5
## 2 ENST00000450305.2 ENSG00000223972.5
## 3 ENST00000473358.1 ENSG00000243485.5
## 4 ENST00000469289.1 ENSG00000243485.5
## 5 ENST00000607096.1 ENSG00000284332.1
## 6 ENST00000606857.1 ENSG00000268020.3
```

```
txi <-
  tximport(quant.files,
           type = "salmon",
           tx2gene = tx2gene,
           txOut = F)
# names(txi)
# head(txi$counts)
```

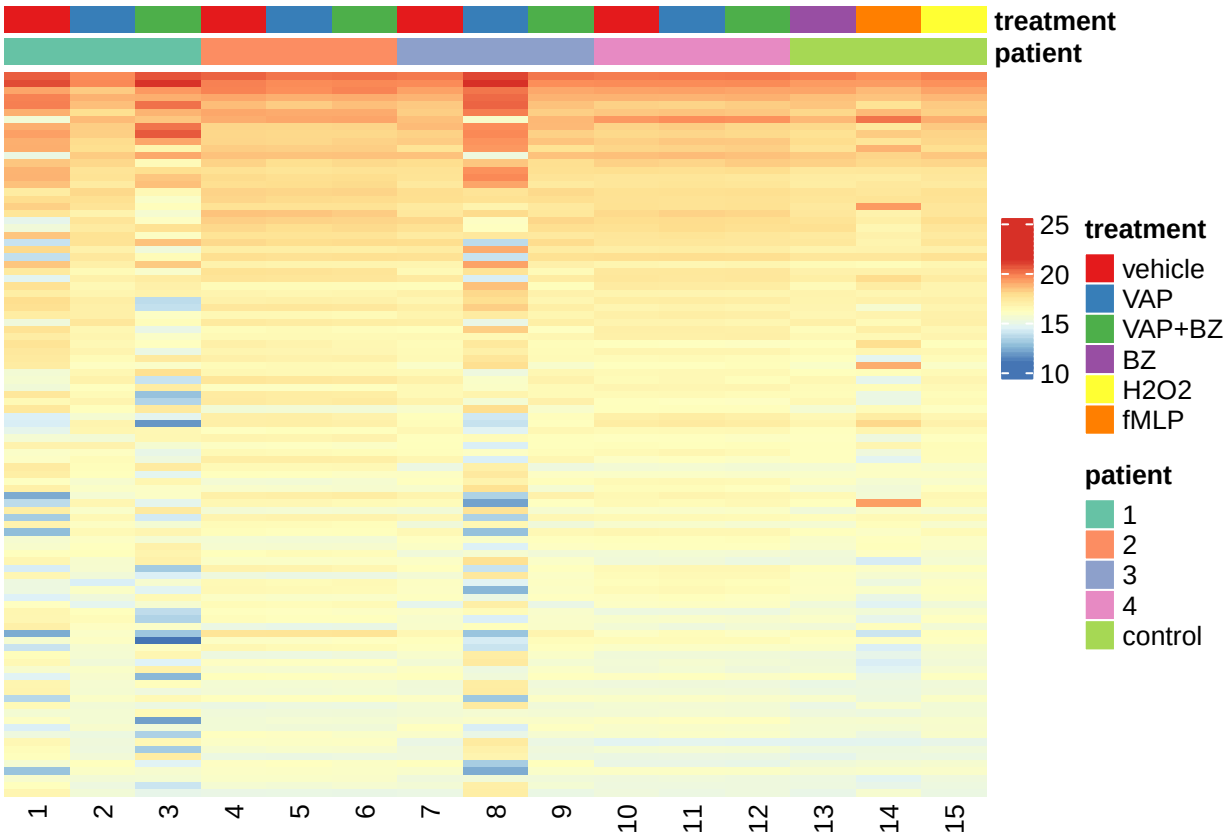
Next, we filtered for genes that are expressed in at least six samples (n = 19240) and perform fitting of the data using the DESeq2 package.

```
expressed <- rowSums(txi$counts > 0) >= 6
# sum(expressed)

txi[1:3] <- lapply(txi[1:3], function(x) x[expressed == T, ])

dds <- DESeqDataSetFromTximport(txi, colData = colData, design = ~ 1)
dds <- DESeq(dds, fitType = "local")
# local fitType was chosen, because residuals were lower for
  # this fitType compared to the default option.

vsd <- vst(dds, blind = T)
```

2.2. heatmap of top 100 most abundantly expressed genes

To show expression profiles for the most abundantly expressed genes, we plotted the genes with top 100 highest expression values, based on normalised counts. It is apparant that expression profiles for samples c18, c20 and c25, and to a lesser extent sample c31, are quite distinct from the other samples.

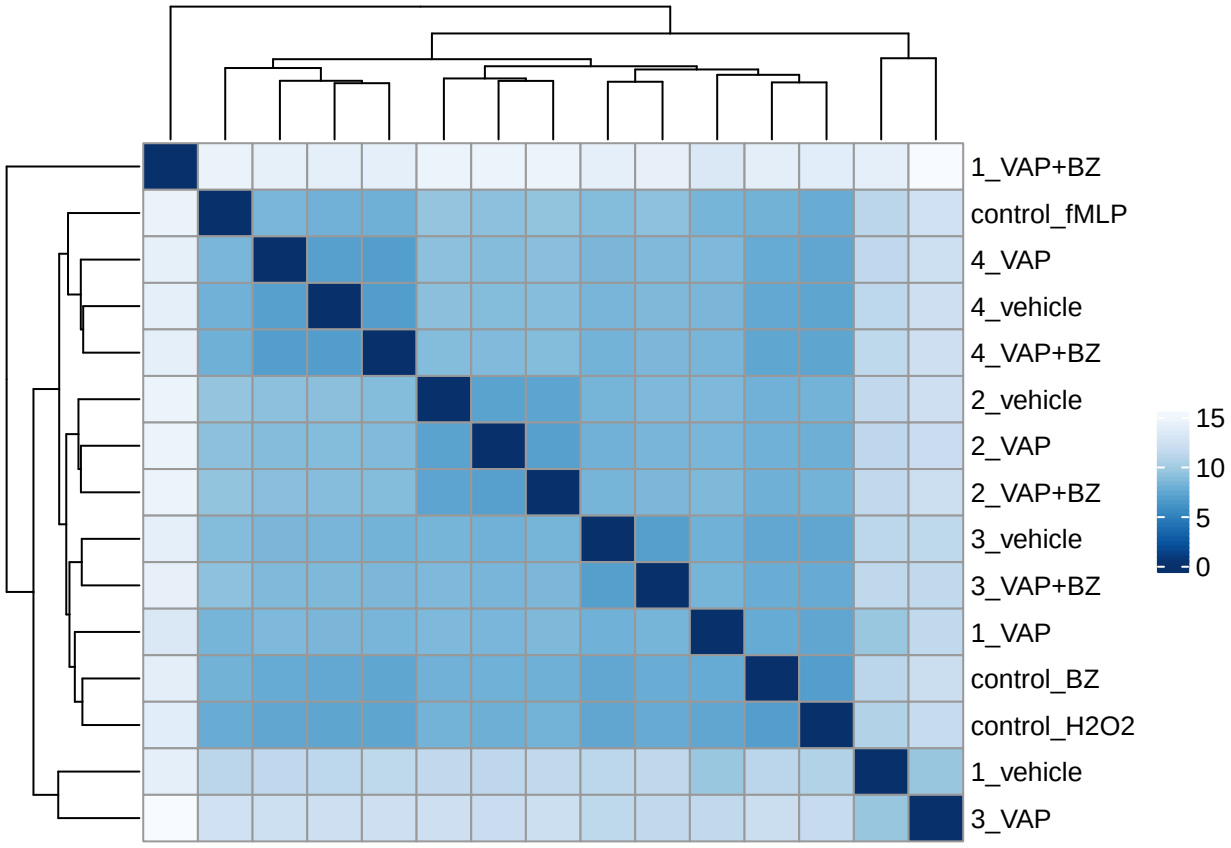
```
library(RColorBrewer)
top100 <- assay(vsd) [order(rowMeans(assay(vsd)), decreasing = T), ] [1:100,]

annoCol <- list(treatment = brewer.pal(6, "Set1"),
                patient = brewer.pal(5, "Set2"))
names(annoCol$treatment) <- unique(colData$treatment)
names(annoCol$patient) <- unique(colData$patient)

pheatmap(top100, cluster_rows = F, cluster_cols = F,
         show_rownames = F,
         annotation_col = colData %>%
           dplyr::select("patient", "treatment"),
         annotation_colors = annoCol)
```



2.3. Heatmap of sample-to-sample euclidean distances

A similar pattern is visible when plotting the euclidean distances between samples. Again, samples c18, c20 and c25 cluster distinctly from the other samples, while samples obtained from volunteers two and four cluster according to donor identity.

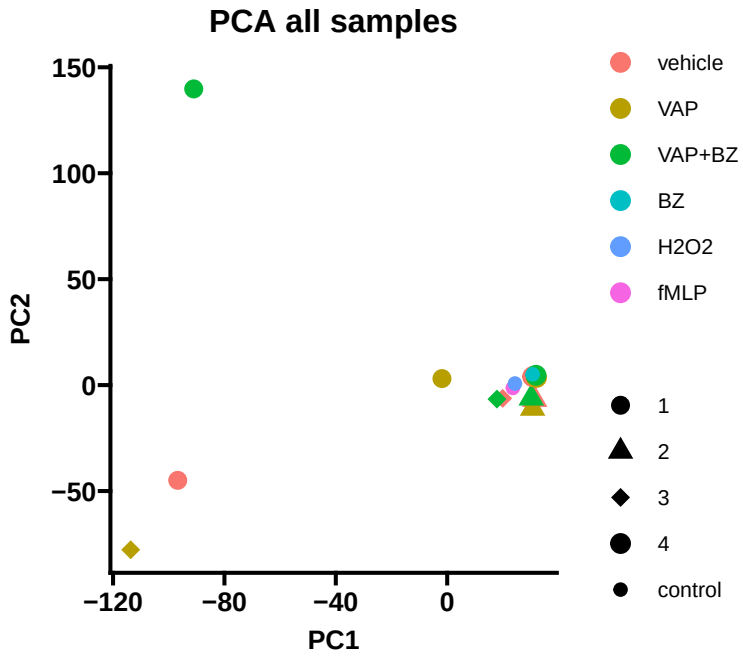
```
sampleDists <- dist(t(log10(assay(vsd)+1)))

library(RColorBrewer)
sampleDistMatrix <- as.matrix(sampleDists)
rownames(sampleDistMatrix) <- paste(dds$patient, dds$treatment, sep = "_")
colnames(sampleDistMatrix) <- NULL
colors <- colorRampPalette(rev(brewer.pal(9, "Blues")))(255)

pheatmap(sampleDistMatrix,
         clustering_distance_rows = sampleDists,
         clustering_distance_cols = sampleDists,
         col = colors)
```



2.4. PCA analysis on normalised expression for all samples

Similarly in principal component analysis, samples c18, c20 and c25 are very distant to the other samples.

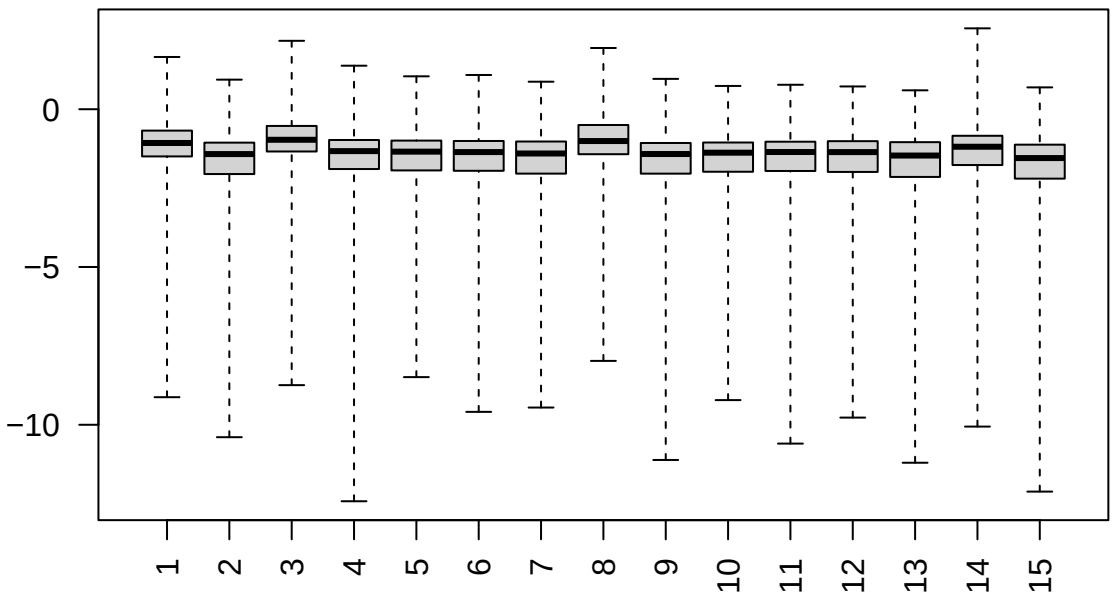
```
pca_counts <- stats::prcomp(t(assay(vsd)), scale = F)

pca_counts$x [,1:2] %>% cbind(colData) %>% as.data.frame() %>%
  ggplot(aes(x = PC1, y = PC2, colour = treatment, shape = patient )) +
  geom_point(size = 3) +
  scale_shape_manual(values = c(16:20)) +
  labs(title = "PCA all samples") +
  theme_prism(base_size = 10) +
  theme(legend.position = "right")
```



2.5. Plotting cook's distances for all samples

When plotting the gene-wise cooks' distances per sample as a measure of fit to the DESeq2 model, samples c18, c20 and c25 show considerably higher values, confirming their status as outliers.

```
boxplot(log10(assays(dds)[["cooks"]]), range=0, las=2)
```

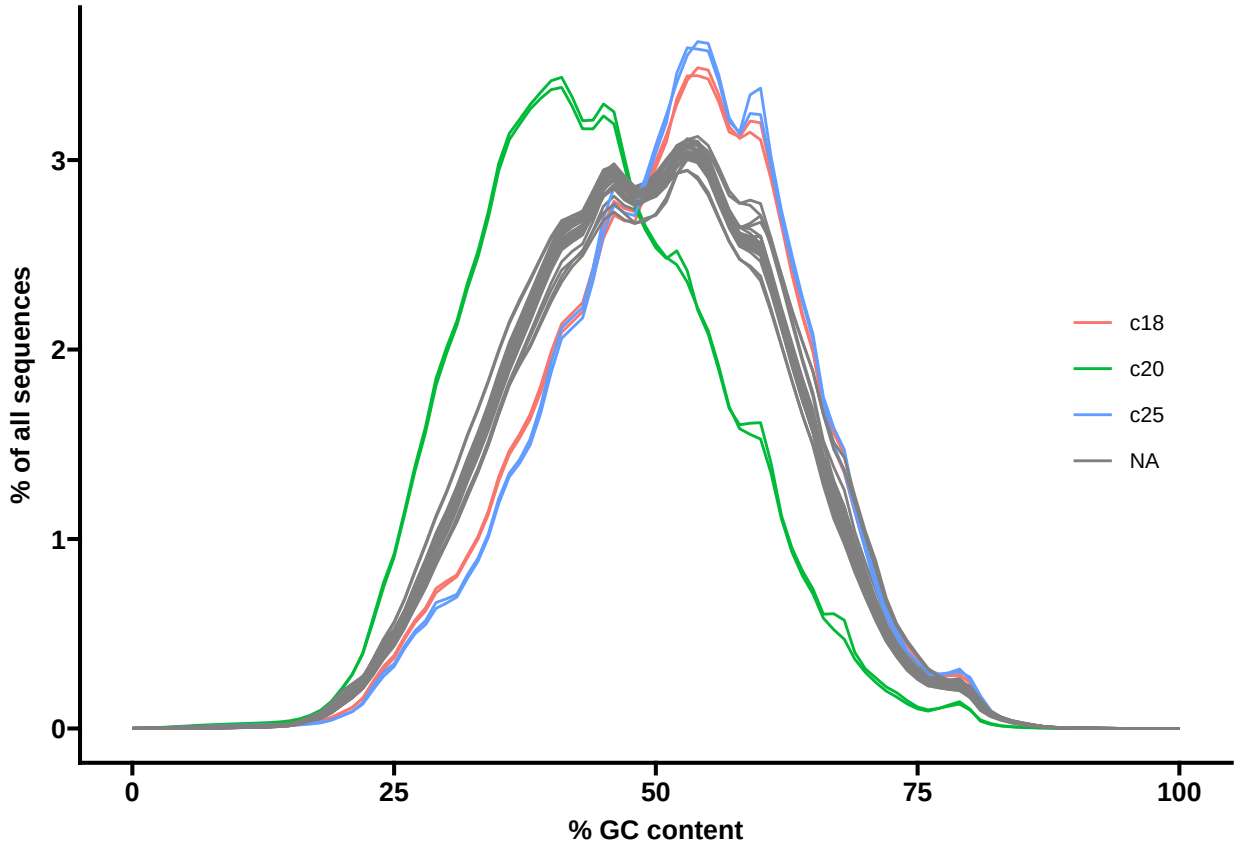


2.6. GC content distribution across samples

A possible reason for the variability in samples might be related to sequencing quality. As can be seen in the figure below, samples c18, c20 and c25 showed a considerable shift in GC content distribution compared to the remainder of the samples, which probably explains why they are outliers.

```
qc_files <-
  c(
    list.files("fastqc", pattern = "1_fastqc.zip", full = T),
    list.files("fastqc", pattern = "2_fastqc.zip", full = T)
  )

qc <-
  qc_read_collection(qc_files,
                     sample_names = word(qc_files, 2, sep = "/"),
                     modules = "Per sequence GC content")

gc_df <- qc$per_sequence_gc_content %>%
  mutate(file = sample, sample = substr(file, 10, 12)) %>%
  mutate(annotation = case_when(
    sample == "c18" ~ "c18",
    sample == "c20" ~ "c20",
    sample == "c25" ~ "c25"
  ))
```

```r
gc_df <- gc_df %>% pivot_wider(
  id_cols = c("GC Content"),
  names_from = "file",
  values_from = "Count"
)

gc_df [, 2:31] <-
  apply(gc_df [, 2:31], 2, function(x)
    x / sum(x, na.rm = T) * 100)

gc_df <-
  gc_df %>% pivot_longer(cols = 2:31,
                         names_to = "file",
                         values_to = "Percent") %>%
  mutate(sample = substr(file, 10, 12)) %>%
  mutate(annotation = case_when(
    sample == "c18" ~ "c18",
    sample == "c20" ~ "c20",
    sample == "c25" ~ "c25"
  ))

gc_df %>%
  ggplot(aes(
    x = `GC Content`,
    y = Percent,
    group = file,
    colour = annotation
  )) +
  geom_line() +
  labs(x = "% GC content", y = "% of all sequences") +
  theme_prism(base_size = 10) +
  theme(legend.position = c(0.9, 0.5))
```



3. Quality control for patients 2 and 4

3.1. PCA analysis after removing samples from patients 1 and 3

To avoid an unbalanced design, all samples from the affected volunteer donors 1 and 3 are removed from subsequent analyses. Again we filter for genes that are expressed in a minimum number of samples (two samples in this case).

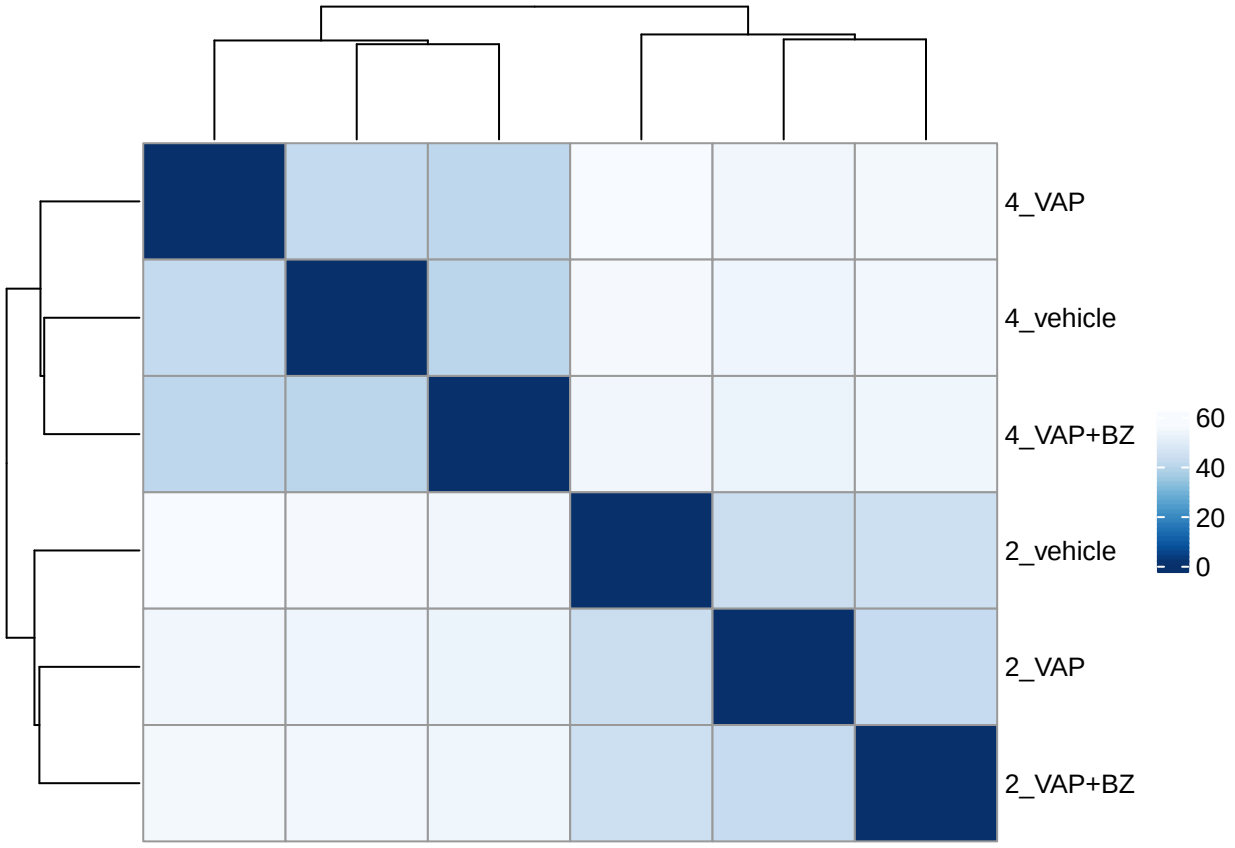
```
txi <-
  tximport(quant.files[-c(1:3, 7:9, 13:15)],
           type = "salmon",
           tx2gene = tx2gene,
           txOut = F)
# names(txi)
# head(txi$counts)

colData <- read_csv("colData.csv") %>% as.data.frame()
rownames(colData) <- colData$ID
colData$treatment <-
  factor(colData$treatment,
         levels = c("vehicle", "VAP", "VAP+BZ", "BZ", "H2O2", "fMLP"))
colData <- colData [-c(1:3, 7:9, 13:15), ]

expressed <- rowSums(txi$counts > 0) >= 2
# sum(expressed)
```

```
txi[1:3] <- lapply(txi[1:3], function(x)
  x[expressed == T,])

dds <-
  DESeqDataSetFromTximport(txi, colData = colData,
                           design = ~ treatment + patient)
dds <- DESeq(dds, fitType = "local")
vsd <- vst(dds, blind = T, fitType = "local")
```

3.2. Heatmap of sample-to-sample euclidean distances

Again, samples cluster according to donor identity but not treatment.

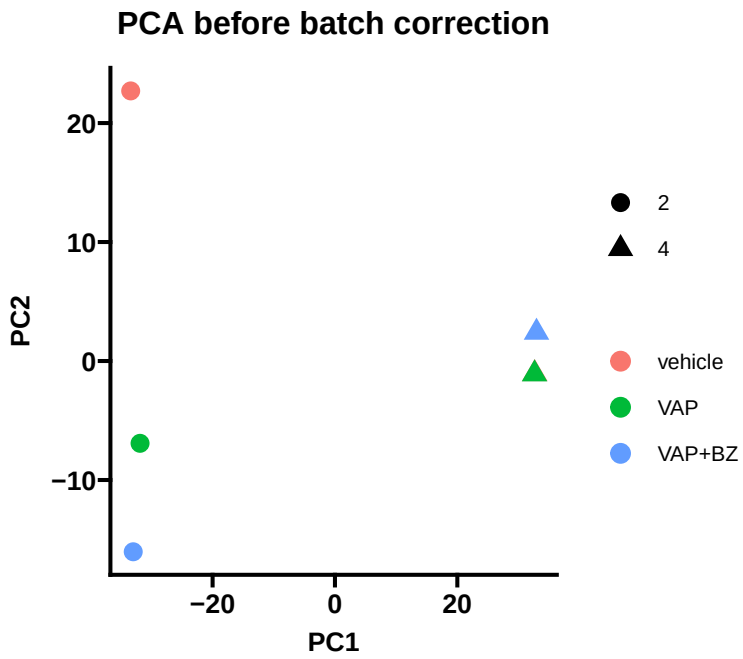
```
sampleDists <- dist(t(log10(counts(dds, normalized = T) + 1)))

library(RColorBrewer)
sampleDistMatrix <- as.matrix(sampleDists)
rownames(sampleDistMatrix) <-
  paste(dds$patient, dds$treatment, sep = "_")
colnames(sampleDistMatrix) <- NULL
colors <- colorRampPalette(rev(brewer.pal(9, "Blues")))(255)

pheatmap(
  sampleDistMatrix,
  clustering_distance_rows = sampleDists,
  clustering_distance_cols = sampleDists,
  col = colors
)
```



3.3. PCA analysis of subset

The same pattern can be observed in PCA which gives a much clearer picture now that the outlier samples have been removed.

```
pca_counts <- stats::prcomp(t(assay(vsd)))

pca_counts$x [, 1:2] %>% cbind(colData) %>% as.data.frame() %>%
  ggplot(aes(
    x = PC1,
    y = PC2,
    colour = treatment,
    shape = patient
  )) +
  geom_point(size = 3) +
  scale_shape_manual(values = 16:18) +
  labs(title = "PCA before batch correction") +
  theme_prism(base_size = 10) +
  theme(legend.position = "right")
```



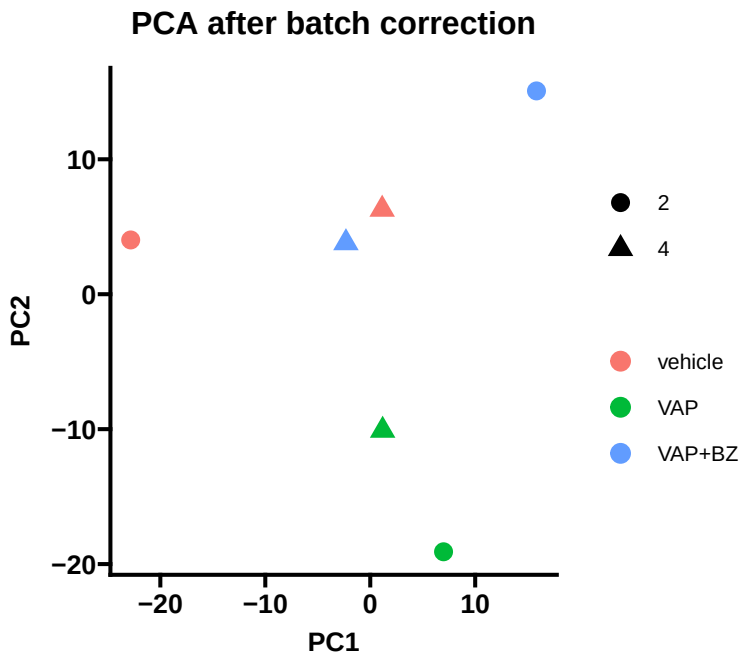
3.4. Testing correction for batch effect

As can be seen from the figure above, variance in this dataset can be mainly explained by donor identity but not treatments. However, we can adjust for donor identity by removing the related batch effect.

```
mm <- model.matrix( ~ treatment, colData)
counts_batch_corr <-
  removeBatchEffect(assay(vsd), design = mm, batch = colData$patient)
```

```
## Coefficients not estimable: treatmentBZ treatmentH2O2 treatmentfMLP
```

```
is_na <- rowSums(is.na(counts_batch_corr)) > 0

pca_corr <- stats::prcomp(t(counts_batch_corr))

pca_corr$x [, 1:2] %>% cbind(colData) %>% as.data.frame() %>%
  ggplot(aes(
    x = PC1,
    y = PC2,
    colour = treatment,
    shape = patient
  )) +
  geom_point(size = 3) +
  scale_shape_manual(values = 16:18) +
  labs(title = "PCA after batch correction") +
  theme_prism(base_size = 10) +
  theme(legend.position = "right")
```



4. Differential gene expression and GSEA

Now that we have established which samples we can analyse, we can perform differential gene expression analyses. We use the default `DESeq2` workflow, except for setting the fitType to "local" instead of "parametric" which leads to a better fit to the DESeq2 model.

4.1 DGE vehicle vs VAP

First we perform differential gene expression analysis for the comparison *vehicle vs VAP*, including shrinkage of log2-fold changes using the apelgm package.

```
res_veh_VAP <-
  results(
    dds,
    contrast = c("treatment", "VAP", "vehicle"),
    alpha = 0.05,
    independentFiltering = FALSE
  )

res_veh_VAP_shrink <-
  lfcShrink(dds, res = res_veh_VAP, coef = "treatment_VAP_vs_vehicle")
summary(res_veh_VAP_shrink)
```

```
##
## out of 21238 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)       : 25, 0.12%
## LFC < 0 (down)     : 7, 0.033%
## outliers [1]       : 0, 0%
## low counts [2]     : 0, 0%
```

```
## (mean count < 0)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

```
res_veh_VAP_shrink_df <-
  res_veh_VAP_shrink %>% as.data.frame() %>%
  mutate(
    SYMBOL = mapIds(org.Hs.eg.db, substr(rownames(res_veh_VAP_shrink), 1, 15),
                    "SYMBOL", "ENSEMBL"),
    padj2 = p.adjust(pvalue, method = "BH")
  ) %>%
  arrange(padj)
```

As can be seen, the number of differentially expressed genes is very low and effect sizes are small (see also **4.6** and **4.7**).

4.2. GSEA vehicle vs VAP

Despite the small changes related to VAP-1 treatment, we want to see if there is a functional enrichment with up- and downregulated genes. Therefore, we create a ranked list of all genes by calculating `rank = -log10(p.adj) * log2FC` and ranking the genes from largest to smallest values. Based on this ranked list, we are looking for enrichment in Gene Ontology and MSigDB v7.4 databases. Unfortunately but as might be expected, we cannot retrieve any meaningful results.

```
ranks_vehicle_VAP <-
  res_veh_VAP_shrink_df %>% mutate(rank = -log10(padj2) * log2FoldChange) %>%
  mutate(ENSEMBL = substr(rownames(res_veh_VAP_shrink_df), 1, 15)) %>%
  mutate(ENTREZ = mapIds(org.Hs.eg.db, ENSEMBL, "ENTREZID", "ENSEMBL")) %>%
  arrange(desc(rank)) %>% filter(!is.na(rank)) %>%
  distinct(ENSEMBL, .keep_all = T)

geneList_vehicle_VAP <- ranks_vehicle_VAP$rank
names(geneList_vehicle_VAP) <- ranks_vehicle_VAP$ENSEMBL
geneList_vehicle_VAP_KEGG <-
  ranks_vehicle_VAP$rank [!is.na(ranks_vehicle_VAP$ENTREZ)]
names(geneList_vehicle_VAP_KEGG) <-
  ranks_vehicle_VAP$ENTREZ [!is.na(ranks_vehicle_VAP$ENTREZ)]
```

```
library(clusterProfiler)
gmt.files <- list.files(pattern = "gmt")
c2.cp.v7.4.entrez <- read.gmt(gmt.files[1])
c3.tft.v7.4.entrez <- read.gmt(gmt.files[2])

res_gseGO_vehicle_VAP <-
  gseGO(
    geneList_vehicle_VAP,
    ont = "BP",
    OrgDb = org.Hs.eg.db,
    pvalueCutoff = 0.05,
    keyType = "ENSEMBL"
  )
res_gseGO_vehicle_VAP_df <-
```

```
  head(setReadable(res_gseGO_vehicle_VAP, org.Hs.eg.db, keyType = "ENSEMBL"),
       n = Inf)
head(res_gseGO_vehicle_VAP_df)
```

```
## [1] ID              Description     setSize         enrichmentScore
## [5] NES             pvalue          p.adjust        qvalues
## <0 rows> (or 0-length row.names)
```

```
res_gseC2CP_veh_VAP <-
  GSEA(geneList_vehicle_VAP_KEGG,
       TERM2GENE = c2.cp.v7.4.entrez,
       pvalueCutoff = 0.05)
res_gseC2CP_veh_VAP_df <-
  head(setReadable(res_gseC2CP_veh_VAP, org.Hs.eg.db, keyType = "ENTREZID"),
       n = Inf)
head(res_gseC2CP_veh_VAP_df)
```

```
## [1] ID              Description     setSize         enrichmentScore
## [5] NES             pvalue          p.adjust        qvalues
## <0 rows> (or 0-length row.names)
```

```
res_gseC3TFT_veh_VAP <-
  GSEA(geneList_vehicle_VAP_KEGG,
       TERM2GENE = c3.tft.v7.4.entrez,
       pvalueCutoff = 0.05)
res_gseC3TFT_veh_VAP_df <-
  head(setReadable(res_gseC3TFT_veh_VAP, org.Hs.eg.db, keyType = "ENTREZID"),
       n = Inf)
head(res_gseC3TFT_veh_VAP_df)
```

```
## [1] ID              Description     setSize         enrichmentScore
## [5] NES             pvalue          p.adjust        qvalues
## <0 rows> (or 0-length row.names)
```

4.4. DGE vehicle vs VAP+BZ

Next we repeat the same procedure for the *vehicle vs VAP+Bz*, yielding similarly negative results, both in differential gene expression and GSEA.

```
res_veh_VAP.BZ <-
  results(
    dds,
    contrast = c("treatment", "VAP.BZ", "vehicle"),
    alpha = 0.05,
    independentFiltering = FALSE
  )

res_veh_VAP.BZ_shrink <-
  lfcShrink(dds, res = res_veh_VAP.BZ, coef = "treatment_VAP.BZ_vs_vehicle")
summary(res_veh_VAP.BZ_shrink)
```

```
##
## out of 21238 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)       : 13, 0.061%
## LFC < 0 (down)     : 11, 0.052%
## outliers [1]       : 0, 0%
## low counts [2]     : 0, 0%
## (mean count < 0)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

```
res_veh_VAP.BZ_shrink_df <-
  res_veh_VAP.BZ_shrink %>% as.data.frame() %>%
  mutate(
    SYMBOL = mapIds(org.Hs.eg.db, substr(rownames(
      res_veh_VAP.BZ_shrink
    ), 1, 15), "SYMBOL", "ENSEMBL"),
    padj2 = p.adjust(pvalue, method = "BH")
  ) %>%
  arrange(padj)
```

4.5 GSEA vehicle vs VAP+BZ

```
ranks_vehicle_VAP.BZ <-
  res_veh_VAP.BZ_shrink_df %>%
  mutate(rank = -log10(pvalue) * log2FoldChange) %>%
  mutate(ENSEMBL = substr(rownames(res_veh_VAP.BZ_shrink_df), 1, 15)) %>%
  mutate(ENTREZ = mapIds(org.Hs.eg.db, ENSEMBL, "ENTREZID", "ENSEMBL")) %>%
  arrange(desc(rank)) %>% filter(!is.na(rank)) %>%
  distinct(ENSEMBL, .keep_all = T)

geneList_vehicle_VAP.BZ <- ranks_vehicle_VAP.BZ$rank
names(geneList_vehicle_VAP.BZ) <- ranks_vehicle_VAP.BZ$ENSEMBL
geneList_vehicle_VAP.BZ_KEGG <-
  ranks_vehicle_VAP.BZ$rank [!is.na(ranks_vehicle_VAP.BZ$ENTREZ)]
names(geneList_vehicle_VAP.BZ_KEGG) <-
  ranks_vehicle_VAP.BZ$ENTREZ [!is.na(ranks_vehicle_VAP.BZ$ENTREZ)]
```

```
gmt.files <- list.files(pattern = "gmt")
c2.cp.v7.4.entrez <- read.gmt(gmt.files[1])
c3.tft.v7.4.entrez <- read.gmt(gmt.files[2])

res_gseGO_vehicle_VAP.BZ <-
  gseGO(
    geneList_vehicle_VAP.BZ,
    ont = "BP",
    OrgDb = org.Hs.eg.db,
    pvalueCutoff = 0.05,
    keyType = "ENSEMBL"
  )
res_gseGO_vehicle_VAP.BZ_df <-
  head(setReadable(res_gseGO_vehicle_VAP.BZ, org.Hs.eg.db, keyType = "ENSEMBL"),
```

```
      n = Inf)
head(res_gseGO_vehicle_VAP.BZ_df)
```

```
## [1] ID              Description     setSize         enrichmentScore
## [5] NES             pvalue          p.adjust        qvalues
## <0 rows> (or 0-length row.names)
```

```
res_gseC2CP_veh_VAP.BZ <-
  GSEA(geneList_vehicle_VAP.BZ_KEGG,
       TERM2GENE = c2.cp.v7.4.entrez,
       pvalueCutoff = 0.05)
res_gseC2CP_veh_VAP.BZ_df <-
  head(setReadable(res_gseC2CP_veh_VAP.BZ, org.Hs.eg.db, keyType = "ENTREZID"),
       n = Inf)
head(res_gseC2CP_veh_VAP.BZ_df)
```

```
## [1] ID              Description     setSize         enrichmentScore
## [5] NES             pvalue          p.adjust        qvalues
## <0 rows> (or 0-length row.names)
```

```
res_gseC3TFT_veh_VAP.BZ <-
  GSEA(geneList_vehicle_VAP.BZ_KEGG,
       TERM2GENE = c3.tft.v7.4.entrez,
       pvalueCutoff = 0.05)
res_gseC3TFT_veh_VAP.BZ_df <-
  head(setReadable(res_gseC3TFT_veh_VAP.BZ, org.Hs.eg.db, keyType = "ENTREZID"),
       n = Inf)
head(res_gseC3TFT_veh_VAP.BZ_df)
```

```
## [1] ID              Description     setSize         enrichmentScore
## [5] NES             pvalue          p.adjust        qvalues
## <0 rows> (or 0-length row.names)
```

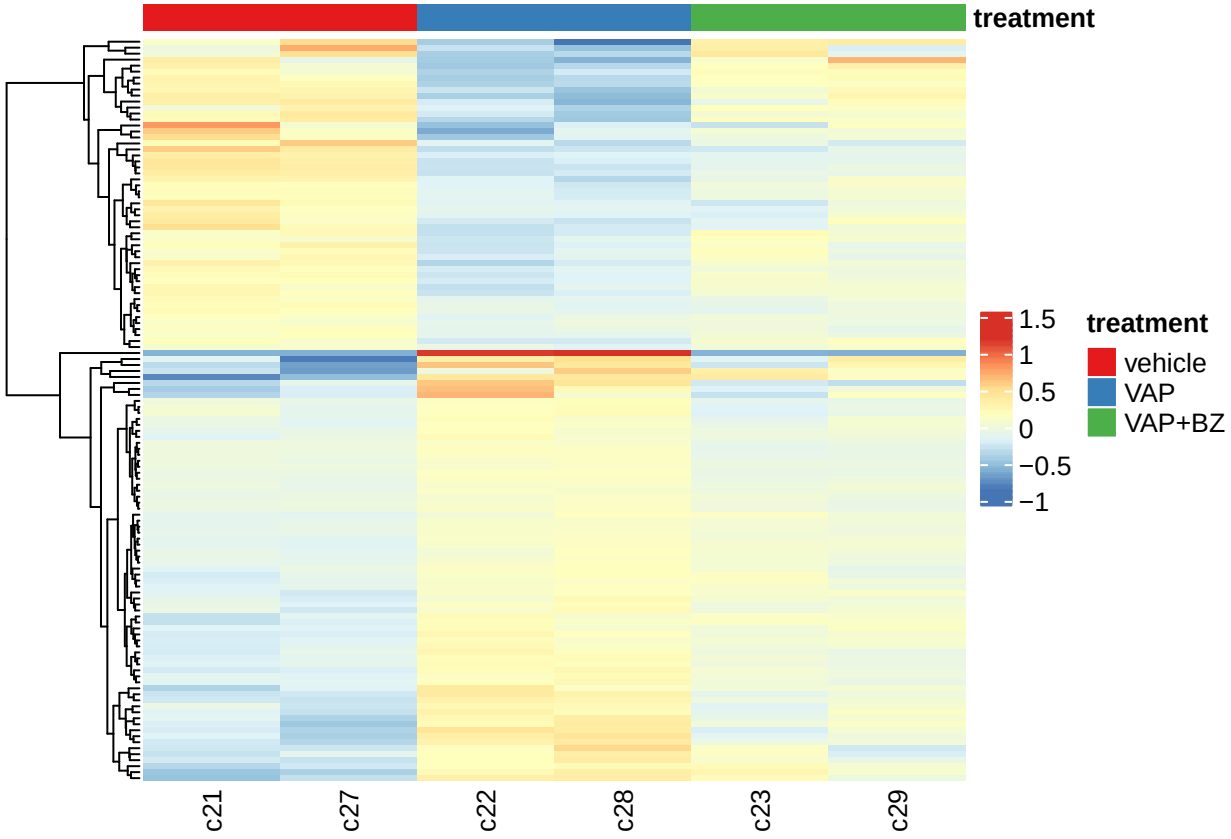
4.6. Heatmap of top 200 differentially expressed genes

For visualisation purposes, we create a heatmap, based on the relative log2 expression values of the top 200 differentially expressed genes, i.e. the 200 genes with lowest adjusted p-values in differential gene expression analysis. Some differences between treatment conditions can be seen but generally log2 fold differences are well below 0.5.

```
top_DEG <-
  rbind(ranks_vehicle_VAP, ranks_vehicle_VAP.BZ) %>% arrange(padj) %>%
  distinct(ENSEMBL, .keep_all = T)

top_DEG <- rownames(top_DEG)[1:200]

vsd <- vst(dds, blind = T, fitType = "local")

mm <- model.matrix( ~ treatment, colData)
vsd_batch_remove <-
  removeBatchEffect(assay(vsd), design = mm, batch = vsd$patient)
```

```
## Coefficients not estimable: treatmentBZ treatmentH2O2 treatmentfMLP
```

```
matrix_log2 <- vsd_batch_remove - rowMeans(vsd_batch_remove)

mat <- matrix_log2 [rownames(matrix_log2) %in% top_DEG, ]
mat <- mat [, order(colData$treatment)]

annoCol <- list(treatment = brewer.pal(3, "Set1"))
names(annoCol$treatment) <- unique(colData$treatment)

pheatmap(
  mat,
  show_rownames = F,
  cluster_cols = F,
  annotation_col = colData %>% dplyr::select(treatment),
  annotation_colors = annoCol
)
```



4.7. Differential gene expression in relation to fMLP as a positive control

Finally, we want to see how the effect size of VAP1 treatment compares to our positive control fMLP. Because DESeq2 does not support the comparison of samples without replicates we need to employ the `edgeR` package. As can be seen in the plot below, fMLP induced higher log-fold changes than VAP-1 (and smaller p-values, but these need to be interpreted with care due to the lack of replicates).

```
txi <-
  tximport(quant.files[c(2, 4, 5, 7, 10, 11, 14)],
           type = "salmon",
           tx2gene = tx2gene,
           txOut = F)

colData <- read_csv("colData.csv") %>% as.data.frame()
rownames(colData) <- colData$ID
colData$treatment <-
  factor(colData$treatment,
         levels = c("vehicle", "VAP", "VAP+BZ", "BZ", "H2O2", "fMLP"))
colData <- colData [c(2, 4, 5, 7, 10, 11, 14), ]
colData$treatment <- droplevels(colData$treatment)

cts <- txi$counts
normMat <- txi$length

# Obtaining per-observation scaling factors for length, adjusted to avoid
# changing the magnitude of the counts.
normMat <- normMat / exp(rowMeans(log(normMat)))
normCts <- cts / normMat

# Computing effective library sizes from scaled counts, to account for
# composition biases between samples.
library(edgeR)
eff.lib <- calcNormFactors(normCts) * colSums(normCts)

# Combining effective library sizes with the length factors, and calculating
# offsets for a log-link GLM.
normMat <- sweep(normMat, 2, eff.lib, "*")
normMat <- log(normMat)

# Creating a DGEList object for use in edgeR.
y <- DGEList(cts, group = colData$treatment)
y <- scaleOffset(y, normMat)
# filtering
keep <- filterByExpr(y)

y <- y[keep,]

y <- calcNormFactors(y)

y <- estimateDisp(y)
y$common.dispersion
```

```
## [1] 0.1836168
```

```
design <- model.matrix( ~ treatment, data = colData)


fit <- glmFit(y, design)
lrt <- glmLRT(fit)
lrt_VAP <- glmLRT(fit, coef = 2)
```

```
topTags(lrt_VAP)
```

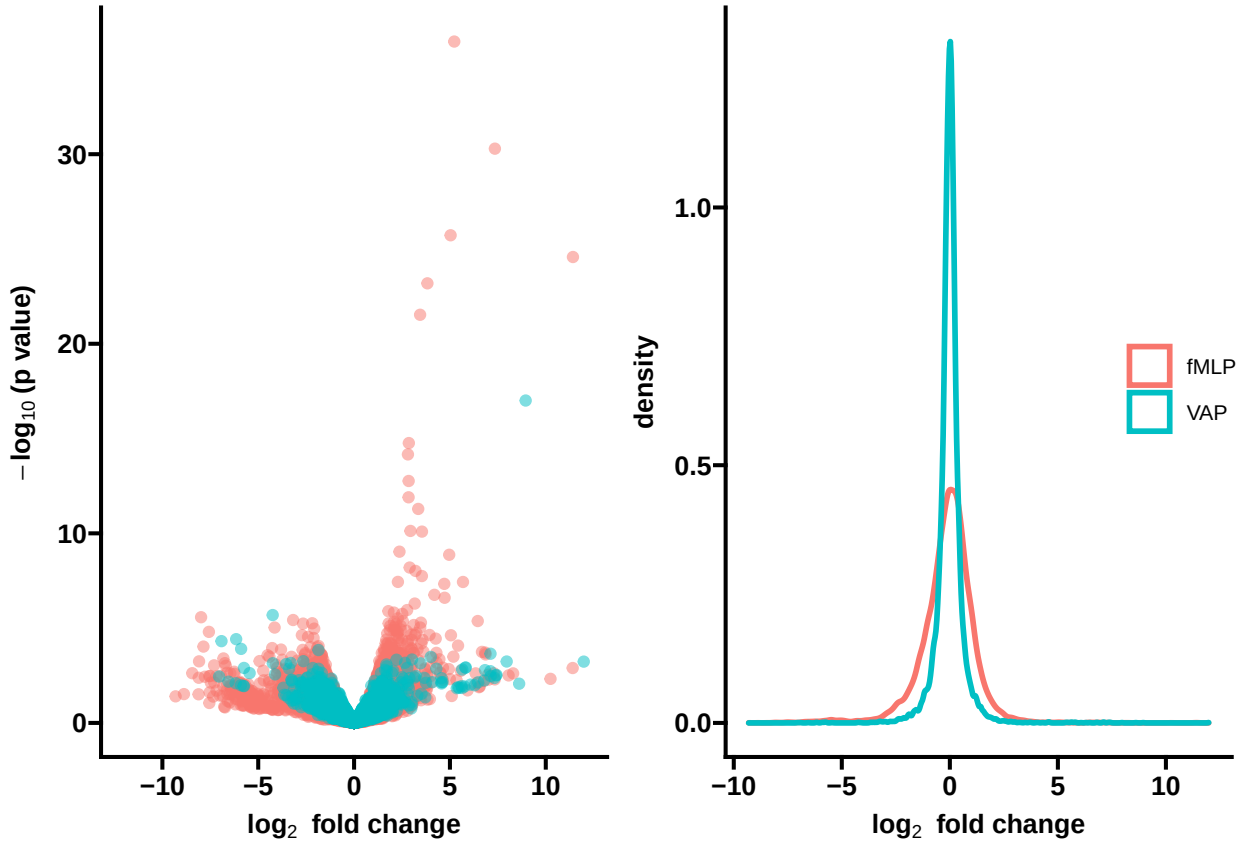
```
## Coefficient:  treatmentVAP
##                         logFC      logCPM       LR       PValue          FDR
## ENSG00000269955.2    8.951909 -0.05220297 73.53665 9.878462e-18 1.399778e-13
## ENSG00000276550.4   -4.246240 -2.13422161 22.55448 2.042678e-06 1.447238e-02
## ENSG00000166317.11  -6.151104 -2.43964641 16.95713 3.823346e-05 1.723416e-01
## ENSG00000261796.1   -6.917344 -0.13205802 16.50001 4.864970e-05 1.723416e-01
## ENSG00000274810.4   -5.892991 -2.64727483 14.72666 1.242764e-04 3.389065e-01
## ENSG00000110693.17  -1.855949 -0.08927826 14.45559 1.435031e-04 3.389065e-01
## ENSG00000212932.3    7.116778 -1.71401775 13.61876 2.239370e-04 4.533124e-01
## ENSG00000259488.2    4.002450 -0.16421580 12.89329 3.297622e-04 5.840913e-01
## ENSG00000139645.9    3.022653 -1.88062047 12.29379 4.544683e-04 5.973198e-01
## ENSG00000166436.16   2.212791 -1.01441116 12.25537 4.639200e-04 5.973198e-01
```

```
topTags(lrt)
```

```
## Coefficient:  treatmentfMLP
##                         logFC    logCPM        LR       PValue          FDR
## ENSG00000170345.9    5.225827 10.973676 159.98276 1.141337e-36 1.617275e-32
## ENSG00000122877.15   7.346816  4.605634 134.13649 5.100566e-31 3.613751e-27
## ENSG00000099985.3    5.035983  9.496911 113.26418 1.888582e-26 8.920404e-23
## ENSG00000120738.7   11.418171  8.339008 108.02857 2.649320e-25 9.385216e-22
## ENSG00000184009.11   3.825786  9.894153 101.69152 6.487753e-24 1.838629e-20
## ENSG00000160888.6    3.444770  9.989685  94.13397 2.948755e-22 6.963977e-19
## ENSG00000159388.5    2.857156 10.694888  63.34977 1.730757e-15 3.503548e-12
## ENSG00000120129.5    2.809285 12.312418  60.64074 6.850157e-15 1.213334e-11
## ENSG00000123689.5    2.847832 11.991687  54.29271 1.727411e-13 2.719712e-10
## ENSG00000141232.4    2.843316  6.208691  50.39901 1.254577e-12 1.777735e-09
```

```
lrt$table$comparison <- "fMLP"
lrt_VAP$table$comparison <- "VAP"

p1 <- rbind(lrt$table, lrt_VAP$table) %>%
  ggplot(aes(
    x = logFC,
    y = -log10(PValue),
    color = comparison
  )) +
  geom_point(alpha = 0.5) +
  scale_x_continuous(limits = c(-12, 12)) +
  labs(x = expression(bold(log [2] ~ " fold change")), y = expression(bold(-log [10] ~
                                                                             "(p value)"))) +
  theme_prism(base_size = 10) +
  theme(legend.position = "none")

p2 <- rbind(lrt$table, lrt_VAP$table) %>%
  ggplot(aes(x = logFC, colour = comparison)) +
  geom_density(alpha = 0.5, size = 1) +
  theme_prism(base_size = 10) +
  labs(x = expression(bold(log [2] ~ " fold change"))) +
  theme(legend.position = c(0.9, 0.5))
```

```
egg::ggarrange(plots = list(p1, p2), ncol = 2)
```




```
sessionInfo()
```

```
## R version 4.1.1 (2021-08-10)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19042)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United Kingdom.1252
## [2] LC_CTYPE=English_United Kingdom.1252
## [3] LC_MONETARY=English_United Kingdom.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United Kingdom.1252
##
## attached base packages:
##  [1] grid      stats4    parallel  stats     graphics  grDevices utils
##  [8] datasets  methods   base
##
## other attached packages:
##  [1] RColorBrewer_1.1-2          fastqcr_0.1.2
##  [3] ComplexHeatmap_2.8.0        clusterProfiler_4.0.5
```

```
##  [5] org.Hs.eg.db_3.13.0         pheatmap_1.0.12
##  [7] factoextra_1.0.7            edgeR_3.34.1
##  [9] limma_3.48.3                DESeq2_1.32.0
## [11] SummarizedExperiment_1.22.0 MatrixGenerics_1.4.3
## [13] matrixStats_0.61.0          tximport_1.20.0
## [15] GenomicFeatures_1.44.2      AnnotationDbi_1.54.1
## [17] Biobase_2.52.0              GenomicRanges_1.44.0
## [19] GenomeInfoDb_1.28.4         IRanges_2.26.0
## [21] S4Vectors_0.30.2            BiocGenerics_0.38.0
## [23] ggprism_1.0.3               forcats_0.5.1
## [25] stringr_1.4.0               dplyr_1.0.7
## [27] purrr_0.3.4                 readr_2.0.2
## [29] tidyr_1.1.4                 tibble_3.1.5
## [31] ggplot2_3.3.5               tidyverse_1.3.1
##
## loaded via a namespace (and not attached):
##   [1] utf8_1.2.2               tidyselect_1.1.1         RSQLite_2.2.8
##   [4] BiocParallel_1.26.2      scatterpie_0.1.7         munsell_0.5.0
##   [7] codetools_0.2-18         withr_2.4.2              colorspace_2.0-2
##  [10] GOSemSim_2.18.1          filelock_1.0.2           highr_0.9
##  [13] knitr_1.36               rstudioapi_0.13          DOSE_3.18.3
##  [16] labeling_0.4.2           bbmle_1.0.24             GenomeInfoDbData_1.2.6
##  [19] polyclip_1.10-0          bit64_4.0.5              farver_2.1.0
##  [22] downloader_0.4           coda_0.19-4              vctrs_0.3.8
##  [25] treeio_1.16.2            generics_0.1.0           xfun_0.26
##  [28] BiocFileCache_2.0.0      R6_2.5.1                 doParallel_1.0.16
##  [31] apeglm_1.14.0            clue_0.3-60              graphlayouts_0.7.1
##  [34] locfit_1.5-9.4           bitops_1.0-7             cachem_1.0.6
##  [37] fgsea_1.18.0             gridGraphics_0.5-1       DelayedArray_0.18.0
##  [40] assertthat_0.2.1         vroom_1.5.5              BiocIO_1.2.0
##  [43] scales_1.1.1             ggraph_2.0.5             enrichplot_1.12.3
##  [46] gtable_0.3.0             egg_0.4.5                Cairo_1.5-12.2
##  [49] tidygraph_1.2.0          rlang_0.4.11             genefilter_1.74.1
##  [52] GlobalOptions_0.1.2      splines_4.1.1            rtracklayer_1.52.1
##  [55] lazyeval_0.2.2           broom_0.7.9              yaml_2.2.1
##  [58] reshape2_1.4.4           modelr_0.1.8             backports_1.2.1
##  [61] qvalue_2.24.0            tools_4.1.1              ggplotify_0.1.0
##  [64] ellipsis_0.3.2           Rcpp_1.0.7               plyr_1.8.6
##  [67] progress_1.2.2           zlibbioc_1.38.0          RCurl_1.98-1.5
##  [70] prettyunits_1.1.1        GetoptLong_1.0.5         viridis_0.6.2
##  [73] cowplot_1.1.1            cluster_2.1.2            haven_2.4.3
##  [76] ggrepel_0.9.1            fs_1.5.0                 magrittr_2.0.1
##  [79] data.table_1.14.2        DO.db_2.9                circlize_0.4.13
##  [82] reprex_2.0.1             mvtnorm_1.1-3            hms_1.1.1
##  [85] patchwork_1.1.1          evaluate_0.14            xtable_1.8-4
##  [88] XML_3.99-0.8             emdbook_1.3.12           readxl_1.3.1
##  [91] shape_1.4.6              gridExtra_2.3            bdsmatrix_1.3-4
##  [94] compiler_4.1.1           biomaRt_2.48.3           crayon_1.4.1
##  [97] shadowtext_0.0.9         htmltools_0.5.2          ggfun_0.0.4
## [100] tzdb_0.1.2               snow_0.4-3               geneplotter_1.70.0
## [103] aplot_0.1.1              lubridate_1.8.0          DBI_1.1.1
## [106] tweenr_1.0.2             dbplyr_2.1.1             MASS_7.3-54
## [109] rappdirs_0.3.3           Matrix_1.3-4             cli_3.0.1
## [112] igraph_1.2.7             pkgconfig_2.0.3          GenomicAlignments_1.28.0
```

```
## [115] numDeriv_2016.8-1.1      xml2_1.3.2              foreach_1.5.1
## [118] ggtree_3.0.4             annotate_1.70.0         XVector_0.32.0
## [121] rvest_1.0.2              yulab.utils_0.0.4       digest_0.6.28
## [124] Biostrings_2.60.2        rmarkdown_2.11          cellranger_1.1.0
## [127] fastmatch_1.1-3          tidytree_0.3.5          restfulr_0.0.13
## [130] curl_4.3.2               Rsamtools_2.8.0         rjson_0.2.20
## [133] lifecycle_1.0.1          nlme_3.1-152            jsonlite_1.7.2
## [136] viridisLite_0.4.0        fansi_0.5.0             pillar_1.6.4
## [139] lattice_0.20-44          KEGGREST_1.32.0         fastmap_1.1.0
## [142] httr_1.4.2               survival_3.2-11         GO.db_3.13.0
## [145] glue_1.4.2               png_0.1-7               iterators_1.0.13
## [148] bit_4.0.4                ggforce_0.3.3           stringi_1.7.5
## [151] blob_1.2.2               memoise_2.0.0           ape_5.5
```