The COVID challenge

Sébastien Wieckowski

2020-06-12

Contents

Introduction

The proposed challenge is about the exploitation of raw data for an extensive RNA-seq exploration of the host response to SARS-CoV-2 and related viruses in different cell lines, tissues and time points, and which include human primary samples as well as *in vivo* studies in ferrets - an incredible dataset from Benjamin TenOever's lab at Mt Sinai! You should check out their recent BioRxiv preprint: SARS-CoV-2 launches a unique transcriptional signature from in vitro, ex vivo, and in vivo systems.

Study design

We first looked at the original study design file, in particular by selecting and sorting the sample, cell_line, treatment (which contains the MOI) and timepoint_postTreatment columns. The 'messy' study design reports a variety of different samples and conditions, including:

- human primary NHBE samples treated mock vs. SARS-CoV-2 (from now SARS), IAV (i.e. influenza A virus), IAVdNS1 or IFN- for different timepoints (12 - 24 h)
- human A549 cell line treated mock vs. SARS (MOIs 0.2 or 2), RSV (MOIs 2 or 15) or HIPV3 (MOI 3) for 24 h
- human A549 cells treated mock vs. IAV (MOI 5) for 9 h
- human A549-ACE2 treated mock vs. SARS (MOIs 0.2, 2) or pretreated with Ruxolitinib and then SARS (MOI 2) for 24 h
- human Calu3 cell line treated mock vs. SARS (MOI 2) for 24 h
- *in vivo* nasal wash and trachea (only one time point) samples from ferrets treated mock vs. SARS (5E4 PFU) for 1, 3, 7 or 14 days, and IAV (1E5 PFU) at day 7 only
- some other lung biopsies from patients

While no gene ontology is available for furret, the time course in the *in vivo* experiment and the comparison with *ex vivo* data using primary normal human bronchial epithelia (NHBE) cells seem to be an interesting starting point for multivariate analysis of the gene signatures regulated during the infection with SARS-CoV-2. Moreover we can compare the responses to the ones induced after infection with influenza A virus.
The plan for the challenge is therefore:

1. classical RNAseq data analysis including filtering, normalization and PCA in both the human primary cells infected with SARS or IAV, and ferret's nasal wash samples
2. DEG in human primary cells up to heatmap and functional enrichment analysis
3. module identification in multiple datasets defined as mock, SARS and IAV during the time course of the infection using `clust`

R packages used

All graphics and data wrangling were handled using the tidyverse suite of packages. All packages used are available from the Comprehensive R Archive Network (CRAN), Bioconductor.org, or Github.

Read the data

The goal of this short script to it make sure we are able to read the count data and the study design for the COVID hackdash into the R environment using tidyverse. Both the human and ferret expression data have been obtained as **raw count tables** from the public GEO entry (see challenge description for more details), therefore no annotation is required.

```
library(tidyverse)

# Reading in study design **as tibble** incl all info for both human and ferret samples
targets <- read_tsv("covid_metadata.txt")

# human covid data read/converted to a matrix with gene symbols as rownames
# initially the dataframe contains X1 col with the name of genes
human_covid_data <- read_tsv("GSE147507_RawReadCounts_Human.tsv")
human_covid_data <- as.matrix(column_to_rownames(human_covid_data, "X1"))

# repeat for the ferret covid data
ferret_covid_data <- read_tsv("GSE147507_RawReadCounts_Ferret.tsv")
ferret_covid_data <- as.matrix(column_to_rownames(ferret_covid_data, "X1"))
# then the datatype is homogeneous in the matrices
```

When we look at the variables of the targets dataframe: sample, description, source, host_species, host_common_name, cell_line, subject_status, tissue_cell_type, treatment, virus_strain, time-point_postTreament, GEO_accession, it seems straightforward to subset data from human and ferret experiments since unique elements from the `host_common_name` variable are human, ferret.

Data filtering

We then prepare and subset data we are interested in for this analysis. In the human experiment we keep the data from primary NHBE samples infected with mock, SARS or IAV (not the other conditions) thanks to the elements in the `tissue_cell_type` variable: primary human bronchial epithelial cells, Lung adenocarcinoma, Lung Biopsy, Nasal Wash, trachea. Of note we pool together the data of mock-treated cells after 12 **and** 24 hours.

```r
targets_human <- targets %>%
  dplyr::select(sample, host_common_name, tissue_cell_type, treatment) %>%
  # timepoint not used here
  dplyr::filter(host_common_name == "human" &
                  tissue_cell_type == "primary human bronchial epithelial cells" &
                  treatment %in% c("Mock treatment", "SARS-CoV-2 infected (MOI 2)",
                                   "IAV infected (MOI 3)"))

# then we don't need the host_common_name and tissue_cell_type anymore
targets_ferret <- targets_human %>%
  dplyr::select(sample, treatment)

treatment_human <- factor(targets_human$treatment,
                          levels = c("Mock treatment", "SARS-CoV-2 infected (MOI 2)",
                                     "IAV infected (MOI 3)"),
                          labels = c("ctrl", "SARS", "IAV"))

# and finally we keep only the columns of interest in the count matrix
human_data <- human_covid_data[, targets_human$sample]
```

Before subsetting the human data, the dimension of the original human_covid_data was 21797, 78and after cleaning, the dimension of the human_data is 21797, 14.

In the *in vivo* study, all the data with nasal wash sample type are kept thanks to the elements in the `tissue_cell_type` variable: primary human bronchial epithelial cells, Lung adenocarcinoma, Lung Biopsy, Nasal Wash, trachea.

```r
# we first select the columns of interest and further filter of ferret data
targets_ferret <- targets %>%
  dplyr::select(sample, host_common_name, tissue_cell_type, treatment, timepoint_postTreament) %>% # be
  dplyr::filter(host_common_name == "ferret" & tissue_cell_type == "Nasal Wash")

# then we don't need the host_common_name and tissue_cell_type anymore
targets_ferret <- dplyr::select(targets_ferret, sample, treatment, timepoint_postTreament)
colnames(targets_ferret) <- c("sample", "treatment", "time")

treatment_ferret <- factor(targets_ferret$treatment,
                          levels = c("Mock treatment", "SARS-CoV-2 infected (5E4 PFU)",
                                     "IAV infected (1E5 PFU)"),
                          labels = c("ctrl", "SARS", "IAV"))
time_ferret <- factor(targets_ferret$time,
                      levels = c("day1", "day3", "day7", "day14"),
                      labels = c(1, 3, 7, 14))

# and finally we keep only the columns of interest in the count matrix
ferret_data <- ferret_covid_data[, targets_ferret$sample]
```

Before subsetting the ferret data, the dimension of the original ferret_covid_data was 32057, 32and after cleaning, the dimension of the ferret_data is 32057, 18.

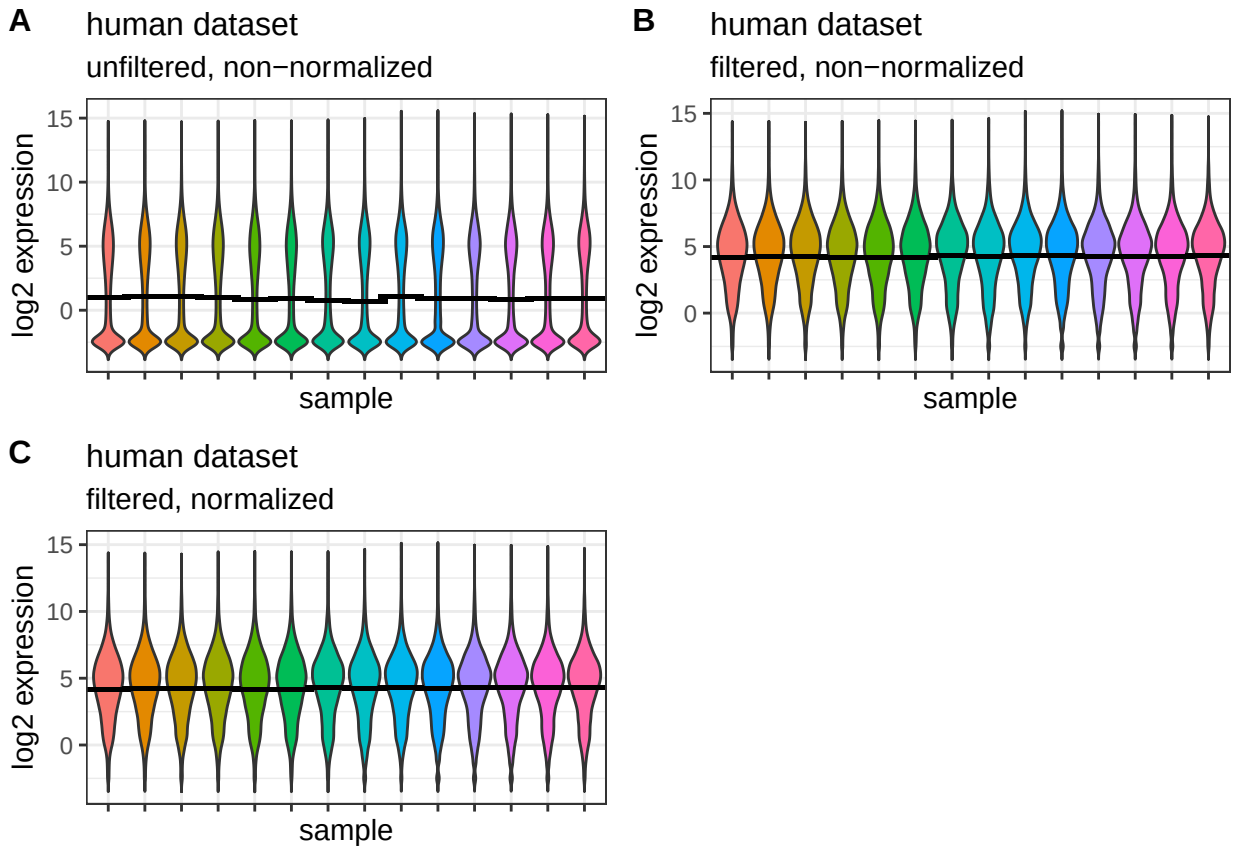
Preprocessing human dataset

Impact of filtering and normalization

```
library(edgeR)
library(matrixStats)

# the function below is used for ploting violins
profile_function <- function(data, samples, title, subtitle) {
    data_count <- edgeR::cpm(data, log = TRUE)
    data_count_df <- tibble::as_tibble(data_count, rownames = "geneID")
    colnames(data_count_df) <- c("geneID", samples)
    data_count_df_pivot <- tidyr::pivot_longer(data_count_df,
                                               cols = -1, # all except col 1
                                               names_to = "samples",
                                               values_to = "expression")

    ggplot(data_count_df_pivot) +
      aes(x = samples, y = expression, fill = samples) +
      geom_violin(trim = FALSE, show.legend = FALSE) +
      stat_summary(fun = "median",
                   geom = "point",
                   shape = 95,
                   size = 10,
                   color = "black",
                   show.legend = FALSE) +
      labs(y = "log2 expression", x = "sample",
           title = title,
           subtitle = subtitle) +
      theme_bw() +
      theme(axis.text.x=element_blank(), # hide x tick labels
            plot.title = element_text(size=12))
}
```

```
library(cowplot)
human_sample_labels <- targets_human$sample
human_DGEList <- DGEList(human_data)
# we take advantage of the profile_function that return violin_plot of cpm
human_p1 <- profile_function(data = human_DGEList,
                             samples = human_sample_labels,
                             title = "human dataset",
                             subtitle = "unfiltered, non-normalized")

human_cpm <- cpm(human_DGEList)
human_keepers <- rowSums(human_cpm > 1) >= 3 # there are triplic/quadrupl-ates
human_DGEList_filtered <- human_DGEList[human_keepers, ]
human_p2 <- profile_function(data = human_DGEList_filtered,
                             samples = human_sample_labels,
                             title = "human dataset",
                             subtitle = "filtered, non-normalized")

human_DGEList_filtered_norm <- calcNormFactors(human_DGEList_filtered,
                                               method = "TMM")
human_p3 <- profile_function(data = human_DGEList_filtered_norm,
                             samples = human_sample_labels,
```

```
                              title = "human dataset",
                              subtitle = "filtered, normalized")

plot_grid(human_p1, human_p2, human_p3,
          labels = c('A', 'B', 'C'),
          label_size = 12)
```



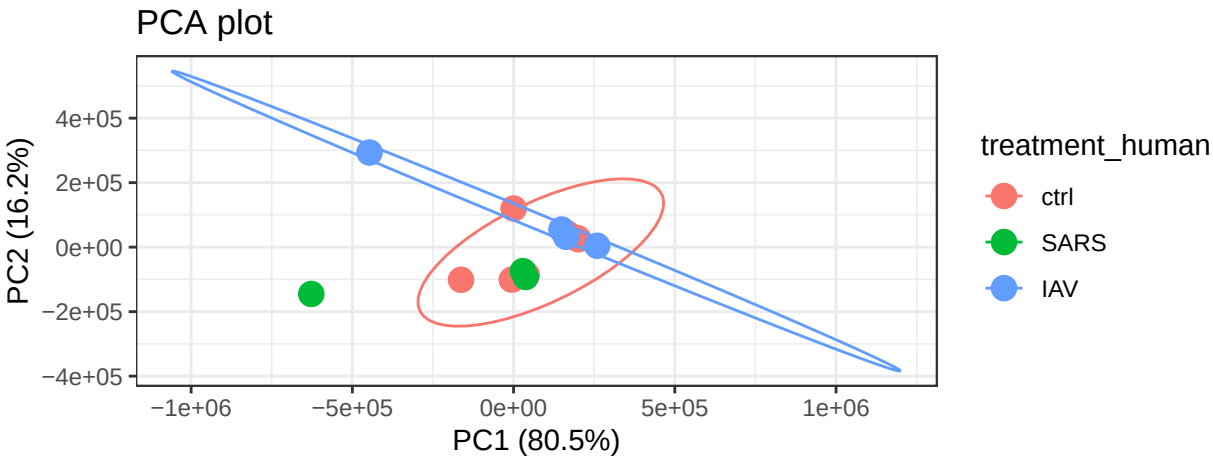
Filtering was carried out to remove lowly expressed genes. Genes with less than 1 count per million (CPM) in at least 5 or more samples filtered out. This reduced the number of genes from 21797 to 12572.

PCA plot

Time to see if infection with SARS-CoV-2 induces a particular gene expression signature, and if this one is different after infection with IAV. The treatment groups have been already factorized, see: ctrl, ctrl, ctrl, SARS, SARS, SARS, ctrl, ctrl, ctrl, ctrl, IAV, IAV, IAV, IAV. *Note: As we don't really need any table for the time being, no datatable is produced in this document.*

```
human_pca_res <- prcomp(t(data.matrix(human_DGEList_filtered_norm)), scale.=F, retx=T)
human_pc_var <- human_pca_res$sdev^2
human_pc_per <- round(human_pc_var/sum(human_pc_var) * 100, 1)
human_pca_res_df <- as_tibble(human_pca_res$x)

ggplot(human_pca_res_df) +
  aes(x = PC1, y = PC2, label = human_sample_labels, color = treatment_human) +
  geom_point(size = 4) +
```

```
  stat_ellipse() +
  xlab(paste0("PC1 (", human_pc_per[1], "%", ")")) +
  ylab(paste0("PC2 (", human_pc_per[2], "%", ")")) +
  labs(title = "PCA plot") +
  coord_fixed() +
  theme_bw()
```



This is not super convincing although PC1 and PC2 account for 96.7% of the total variance.

Differentially expressed genes

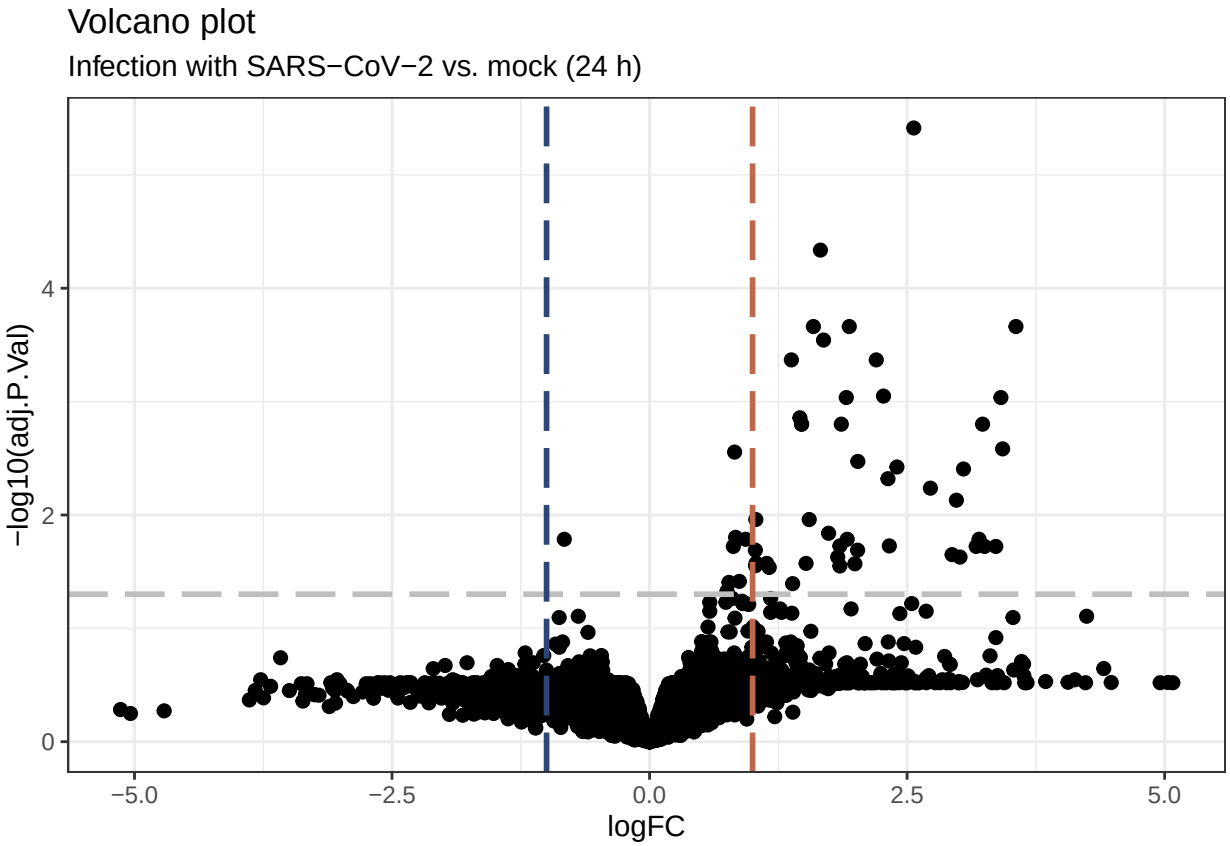
To identify differentially expressed genes, precision weights were first applied to each gene based on its mean-variance relationship using VOOM, then data was normalized using the TMM method in EdgeR. Linear modeling and bayesian stats were employed via Limma to find genes that were up- or down-regulated in leishmania patients by 4-fold or more, with a false-discovery rate (FDR) of 0.01.

Volcano plot

Let's see if there is any interesting gene differentially expressed detected between SARS-CoV-2 and mock conditions.

```
library(limma)
library(edgeR)
library(gt)

design <- model.matrix(~0 + treatment_human)
```

```r
colnames(design) <- levels(treatment_human)

v_human_DGEList_filtered_norm <- voom(human_DGEList_filtered_norm, design, plot = F)
human_fit <- lmFit(v_human_DGEList_filtered_norm, design)
human_contrast_matrix <- makeContrasts(infection = SARS - ctrl,
                                       levels = design)

human_fits <- contrasts.fit(human_fit, human_contrast_matrix)
human_ebFit <- eBayes(human_fits)
human_top_hits <- topTable(human_ebFit,
                           adjust = "BH",
                           coef = 1,
                           number = 20000,
                           sort.by = "logFC")

human_top_hits_df <- human_top_hits %>%
  as_tibble(rownames = "geneID")

ggplot(human_top_hits_df) +
  aes(y = -log10(adj.P.Val), x = logFC, text = paste("Symbol:", geneID)) +
  geom_point(size = 2) +
  geom_hline(yintercept = -log10(0.05), linetype="longdash", colour="grey", size=1) +
  geom_vline(xintercept = 1, linetype="longdash", colour="#BE684D", size=1) +
  geom_vline(xintercept = -1, linetype="longdash", colour="#2C467A", size=1) +
  labs(title="Volcano plot",
       subtitle = "Infection with SARS-CoV-2 vs. mock (24 h)") +
  theme_bw()
```



From the volcano plot, it seems that a dozen of genes are upregulated after infection by SARS-CoV-2.

Table of DEGs

```
human_results <- decideTests(human_ebFit,
                             method = "global",
                             adjust.method = "BH",
                             p.value = 0.05,
                             lfc = 1)
colnames(v_human_DGEList_filtered_norm$E) <- human_sample_labels
human_diffGenes <- v_human_DGEList_filtered_norm$E[human_results[, 1] !=0, ]
human_diffGenes_df <- as_tibble(human_diffGenes, rownames = "geneID")

human_diffGenes_df %>%
  sample_n(10) %>%
  gt() %>%
  fmt_number(columns = 2:11, decimals = 2)
```

geneID	Series1_NHBE_Mock_1	Series1_NHBE_Mock_2	Series1_NHBE_Mock_3	Series1_NHBE_SARS-C
TRIML2	0.94	1.23	0.84	
COL8A1	5.42	5.26	5.29	
IL32	5.45	5.30	5.08	
DTX2	6.19	6.36	5.89	
C1QTNF1	1.93	1.76	1.90	
TNFSF14	−0.40	0.47	−0.04	

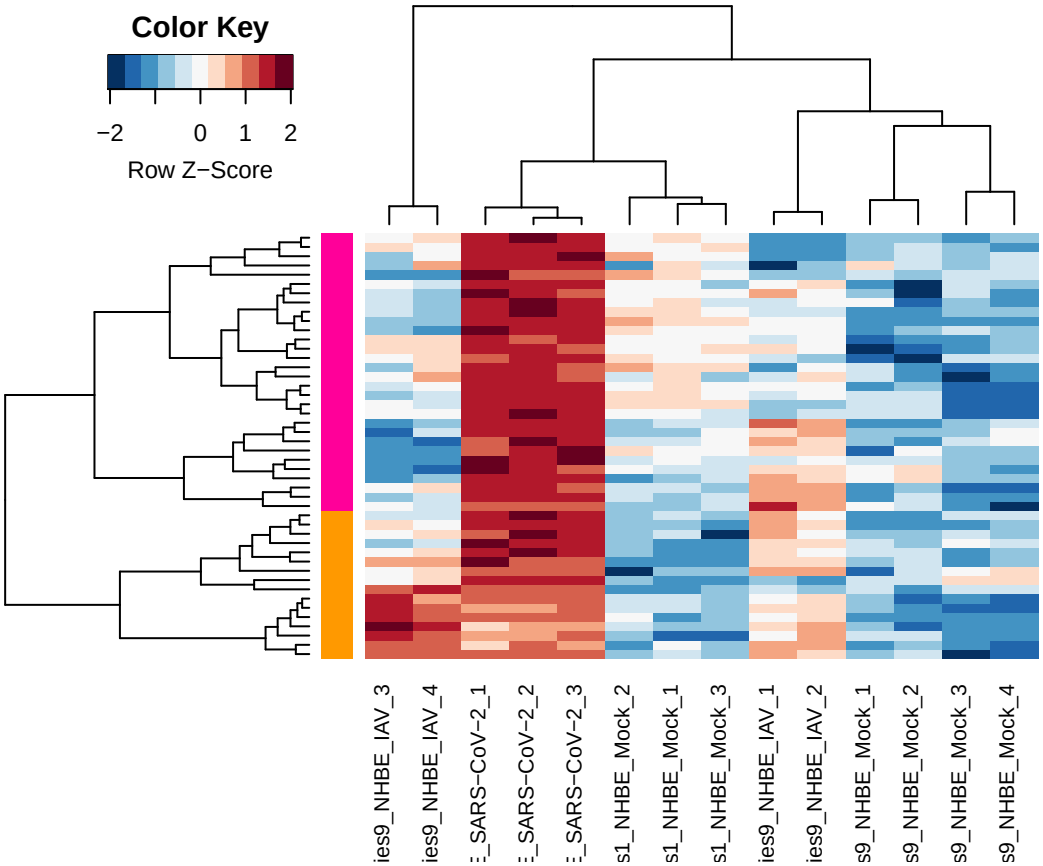
IL6	1.40	1.34	0.20
BPGM	5.37	5.27	5.53
OAS3	5.42	5.57	5.19
MMP9	2.93	3.33	2.79

Great to see IL-6 in the list, but nothing obvious otherwise...

Heatmaps and modules

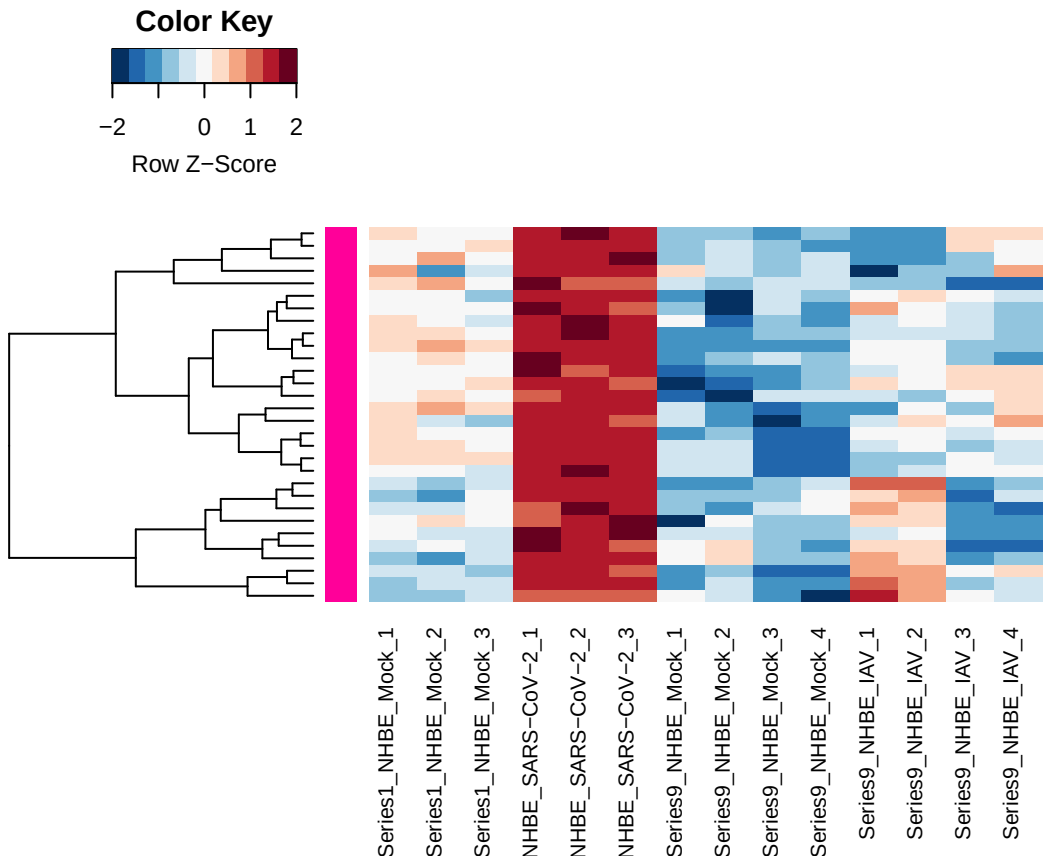
Pearson correlation was used to cluster **46** differentially expressed genes, which were then represented as heatmap with the data scaled by Zscore for each row.

```
library(gplots)
library(RColorBrewer)

myheatcolors <- rev(brewer.pal(name="RdBu", n=11))

clustRows <- hclust(as.dist(1 - cor(t(human_diffGenes),
                                    method = "pearson")),
                    method = "complete")
clustColumns <- hclust(as.dist(1 - cor(human_diffGenes,
                                       method="spearman")),
                       method="complete")

module.assign <- cutree(clustRows, k = 2)
module.color <- rainbow(length(unique(module.assign)), start=0.1, end=0.9)
module.color <- module.color[as.vector(module.assign)]
heatmap.2(human_diffGenes,
          Rowv = as.dendrogram(clustRows),
          Colv = as.dendrogram(clustColumns),
          RowSideColors = module.color,
          col = myheatcolors, scale = 'row', labRow = NA,
          density.info = "none", trace = "none",
          cexRow = 1, cexCol = 1, margins = c(8,8))
```



It looks like most of the genes are upregulated upon infection with SARS-CoV-2. Interestinlgy infection with IAV also lead to a similar profile but to a lesser extent. We then focus the rest of the analysis with upregulated genes. Of note, some replicate in the IAV treatment group have kind of opposite profile, which fits well with the PCA analysis.

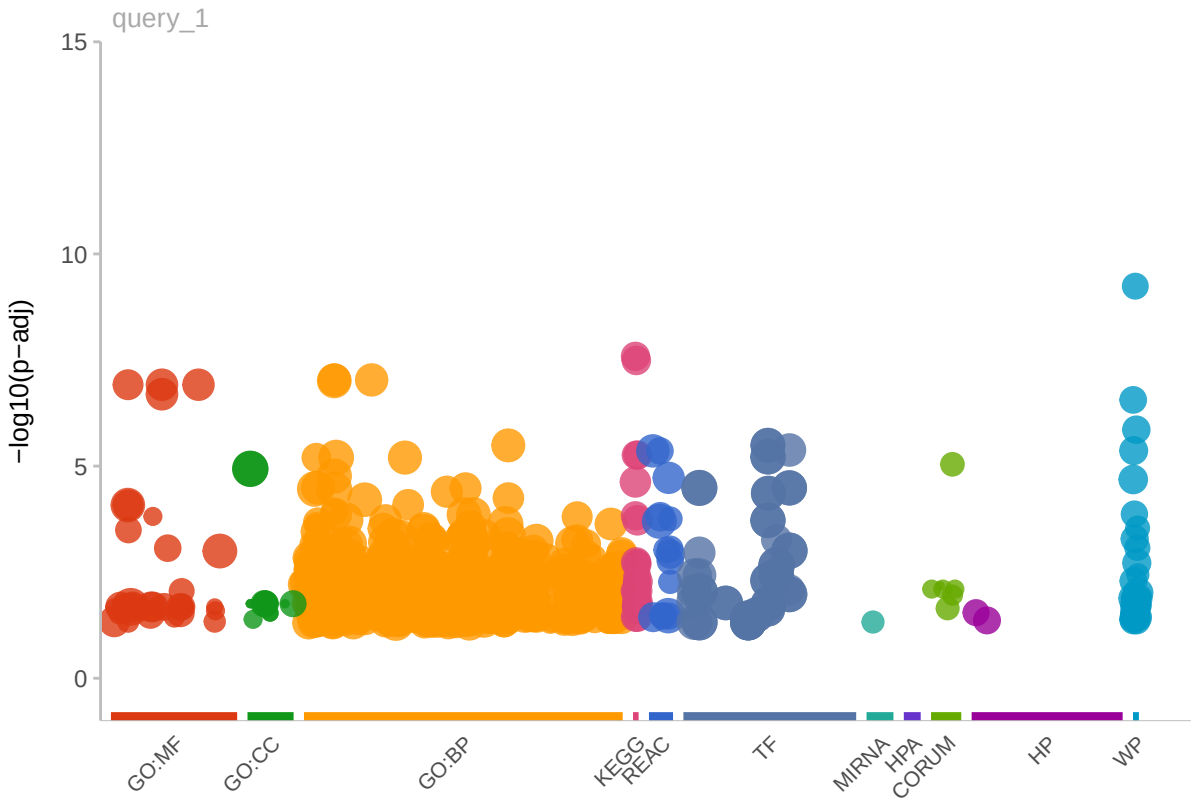
```
modulePick <- 2
myModule_up <- human_diffGenes[names(module.assign[module.assign %in% modulePick]), ]
hrsub_up <- hclust(as.dist(1 - cor(t(myModule_up),
                                   method = "pearson")),
                   method = "complete")

heatmap.2(myModule_up,
          Rowv = as.dendrogram(hrsub_up),
          Colv = NA,
          labRow = NA,
          col = myheatcolors, scale = "row",
          density.info = "none", trace = "none",
          RowSideColors = module.color[module.assign %in% modulePick],
          margins=c(10,8))
```



GO enrichment

GO enrichment for the 30 genes induced by infection with SARS-Cov-2

```
library(GSEABase) # functions and methods for Gene Set Enrichment Analysis
library(Biobase) # base functions for bioconductor; required by GSEABase
library(GSVA) # GSVA, a non-parametric and unsupervised method for estimating variation of gene set enr
library(gprofiler2) # tools for accessing the GO enrichment results using g:Profiler web resources
library(clusterProfiler) # provides a suite of tools for functional enrichment analysis
gost.res_up <- gost(rownames(myModule_up),
                    organism = "hsapiens",
                    correction_method = "fdr")
gostplot(gost.res_up,
         interactive = F,
         capped = F) # set interactive=FALSE to get plot for publications
```



Most of the upregulated genes lies within the **cellular component** ontology group. It might also be interesting to see more closely what are the genes in the WP group.

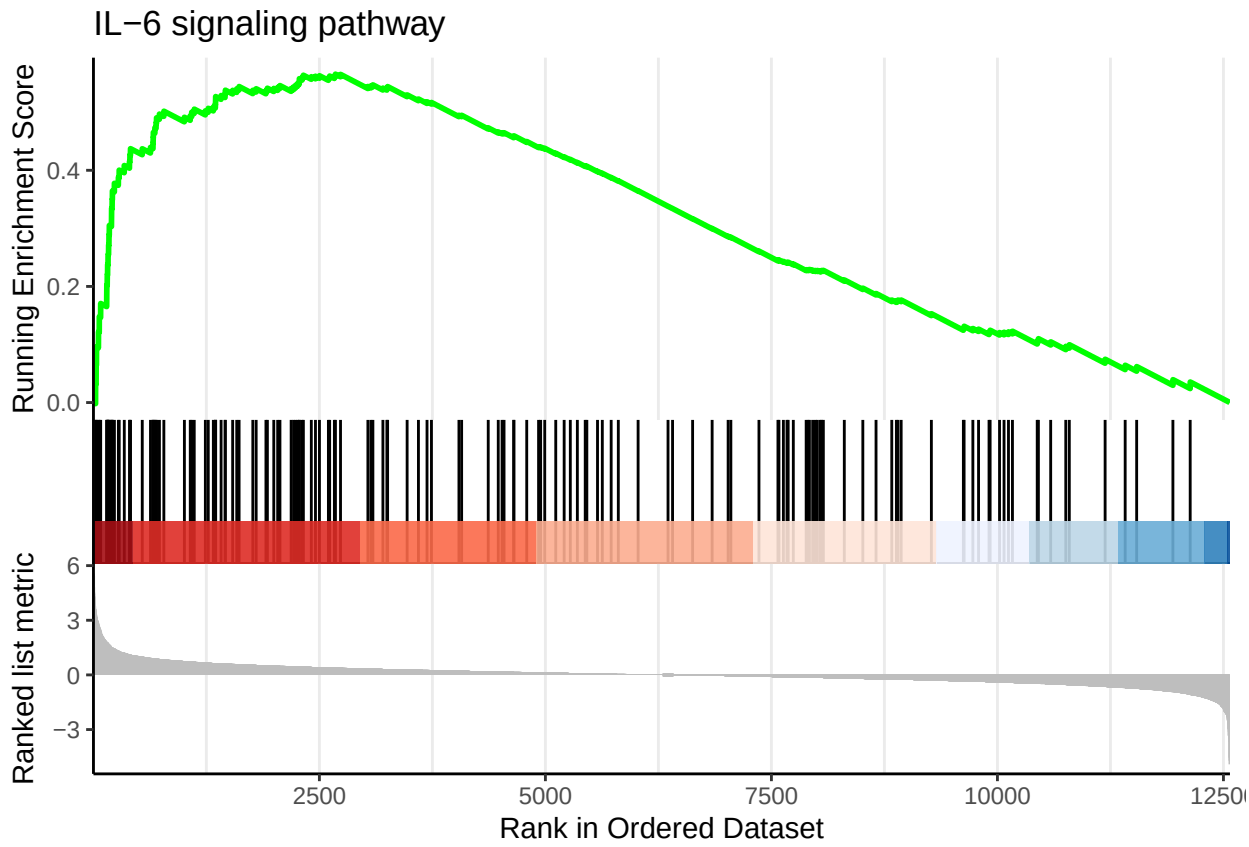
GSEA

```
library(msigdbr) # access to msigdb collections directly within R
library(enrichplot) # great for making the standard GSEA enrichment plots

hs_gsea_c2 <- msigdbr(species = "Homo sapiens",
                      category = "C2") %>%
  dplyr::select(gs_name, gene_symbol) # just get the columns corresponding to signature name and gene s

# Now that you have your msigdb collections ready, prepare your data
# Pull out just the columns corresponding to gene symbols and LogFC for at least one pairwise compariso
log2.cpm.filtered.norm.df <- cpm(human_DGEList_filtered_norm, log = TRUE) %>%
  as_tibble(rownames = "geneID")
# needs to rename columns
colnames(log2.cpm.filtered.norm.df) <- c("geneID",
                                         c("mock_1", "mock_2", "mock_3",
                                           "sars_1", "sars_2", "sars_3",
                                           "mock_4", "mock_5", "mock_6", "mock_7",
                                           "iav_1", "iav_2", "iav_3", "iav_4"))
mydata.df <- log2.cpm.filtered.norm.df  %>%
  mutate(mock.AVG = (mock_1 + mock_2 + mock_3 + mock_4 + mock_5 + mock_6 + mock_7) / 7,
         infect.AVG = (sars_1 + sars_2 + sars_3) / 3,
         #now make columns comparing each of the averages above that you're interested in
```

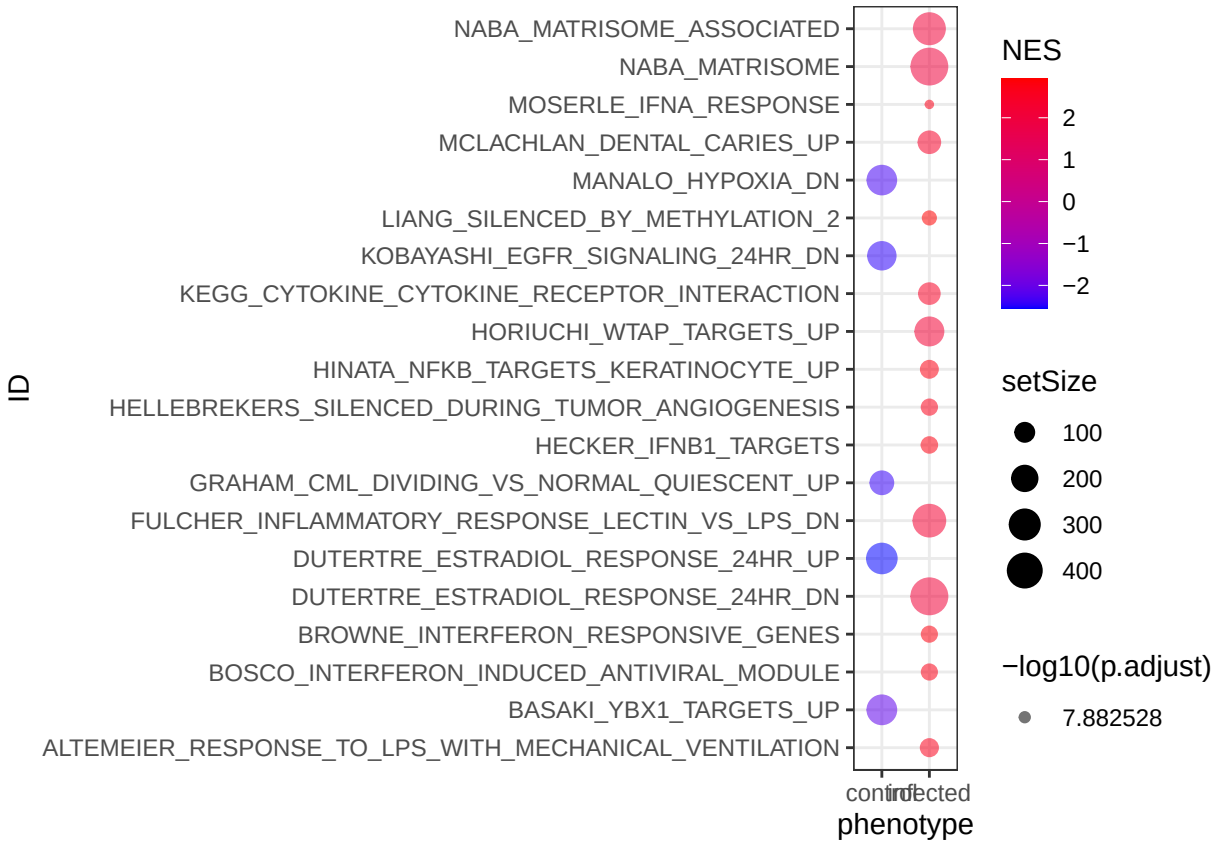
```
        LogFC = (infect.AVG - mock.AVG)) %>%
  mutate_if(is.numeric, round, 2)
# Pull out just the columns corresponding to gene symbols and LogFC for at least one pairwise comparisо
mydata.df.sub <- dplyr::select(mydata.df, geneID, LogFC)
# construct a named vector
mydata.gsea <- mydata.df.sub$LogFC
names(mydata.gsea) <- as.character(mydata.df.sub$geneID)
mydata.gsea <- sort(mydata.gsea, decreasing = TRUE)


mydata.df.sub <- dplyr::select(mydata.df, geneID, LogFC)
mydata.gsea <- mydata.df.sub$LogFC
names(mydata.gsea) <- as.character(mydata.df.sub$geneID)
mydata.gsea <- sort(mydata.gsea, decreasing = TRUE)

# run GSEA using the 'GSEA' function from clusterProfiler
myGSEA.res <- GSEA(mydata.gsea,
                   TERM2GENE = hs_gsea_c2,
                   verbose=FALSE)
myGSEA.df <- as_tibble(myGSEA.res@result)

# view results as a table
myGSEA.df %>%
  dplyr::arrange(NES) %>%
  sample_n(10) %>%
  gt()
```

ID	Description	set
HELLER_SILENCED_BY_METHYLATION_UP	HELLER_SILENCED_BY_METHYLATION_UP	1
SENESE_HDAC1_TARGETS_UP	SENESE_HDAC1_TARGETS_UP	3
RASHI_RESPONSE_TO_IONIZING_RADIATION_2	RASHI_RESPONSE_TO_IONIZING_RADIATION_2	1
HSIAO_LIVER_SPECIFIC_GENES	HSIAO_LIVER_SPECIFIC_GENES	1
ONDER_CDH1_TARGETS_2_UP	ONDER_CDH1_TARGETS_2_UP	2
CHIBA_RESPONSE_TO_TSA_DN	CHIBA_RESPONSE_TO_TSA_DN	2
ROZANOV_MMP14_TARGETS_UP	ROZANOV_MMP14_TARGETS_UP	1
JOHNSTONE_PARVB_TARGETS_2_UP	JOHNSTONE_PARVB_TARGETS_2_UP	1
YAGUE_PRETUMOR_DRUG_RESISTANCE_DN	YAGUE_PRETUMOR_DRUG_RESISTANCE_DN	1
TSAI_DNAJB4_TARGETS_UP	TSAI_DNAJB4_TARGETS_UP	1

```
# create enrichment plots using the enrichplot package
gseaplot2(myGSEA.res,
          geneSetID = 21, # IL-6 signaling pathway
          pvalue_table = FALSE, #can set this to FALSE for a cleaner plot
          title = "IL-6 signaling pathway")
```




```
# add a variable to this result that matches enrichment direction with phenotype
myGSEA.df <- myGSEA.df %>%
  mutate(phenotype = case_when(
    NES > 0 ~ "infected",
    NES < 0 ~ "control"))

# create 'bubble plot' to summarize y signatures across x phenotypes
ggplot(myGSEA.df[1:20,],
       aes(x=phenotype, y=ID)) +
  geom_point(aes(size=setSize,
                 color = NES,
                 alpha = -log10(p.adjust))) +
  scale_color_gradient(low = "blue", high = "red") +
  theme_bw()
```

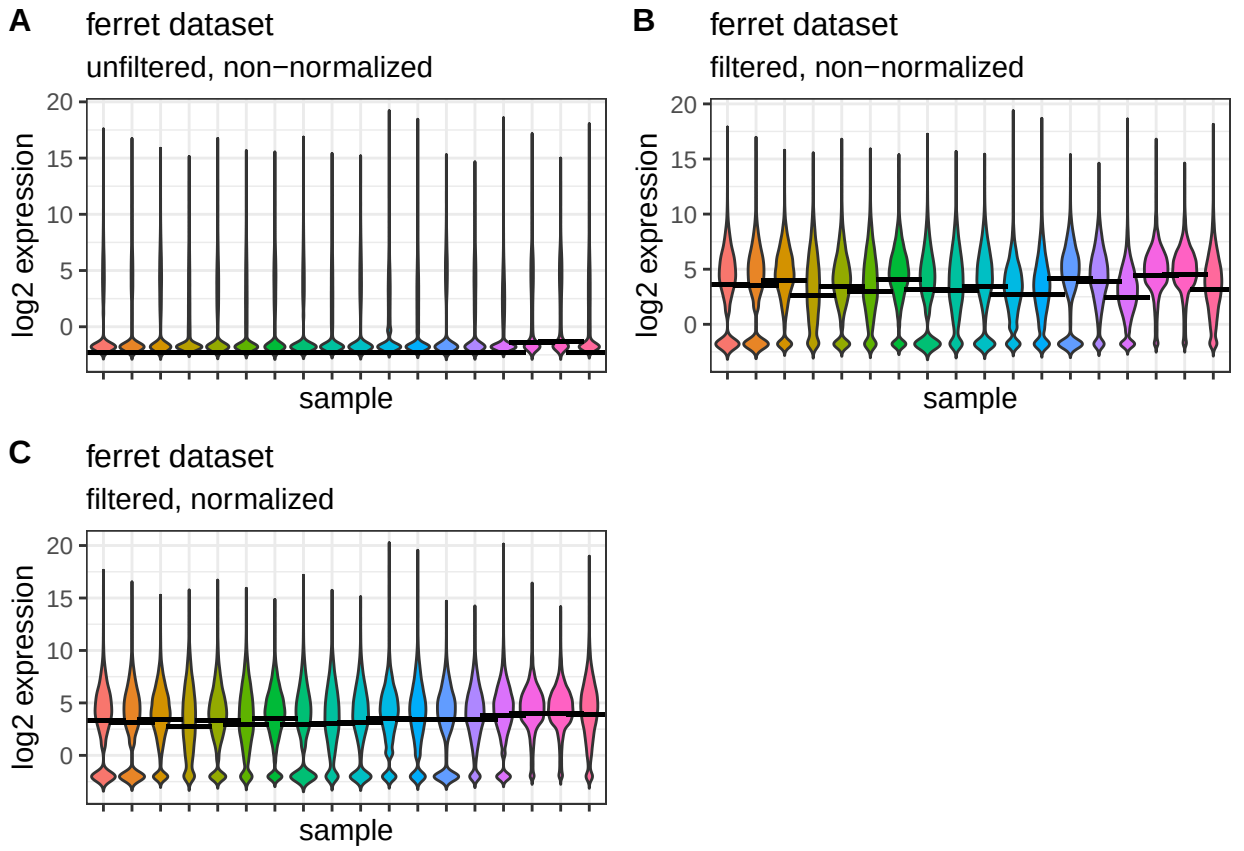


Preprocessing ferret dataset

Impact of filtering and normalization

```
ferret_sample_labels <- targets_ferret$sample
ferret_DGEList <- DGEList(ferret_data)
# we take advantage of the profile_function that return violin_plot of cpm
ferret_p1 <- profile_function(data = ferret_DGEList,
                              samples = ferret_sample_labels,
                              title = "ferret dataset",
                              subtitle = "unfiltered, non-normalized")

ferret_cpm <- cpm(ferret_DGEList)
ferret_keepers <- rowSums(ferret_cpm > 10) >= 2 # all duplicates
ferret_DGEList_filtered <- ferret_DGEList[ferret_keepers, ]
ferret_p2 <- profile_function(data = ferret_DGEList_filtered,
                              samples = ferret_sample_labels,
                              title = "ferret dataset",
                              subtitle = "filtered, non-normalized")

ferret_DGEList_filtered_norm <- calcNormFactors(ferret_DGEList_filtered,
                                                method = "TMM")
ferret_p3 <- profile_function(data = ferret_DGEList_filtered_norm,
                              samples = ferret_sample_labels,
```

```
                              title = "ferret dataset",
                              subtitle = "filtered, normalized")

plot_grid(ferret_p1, ferret_p2, ferret_p3,
          labels = c('A', 'B', 'C'),
          label_size = 12)
```



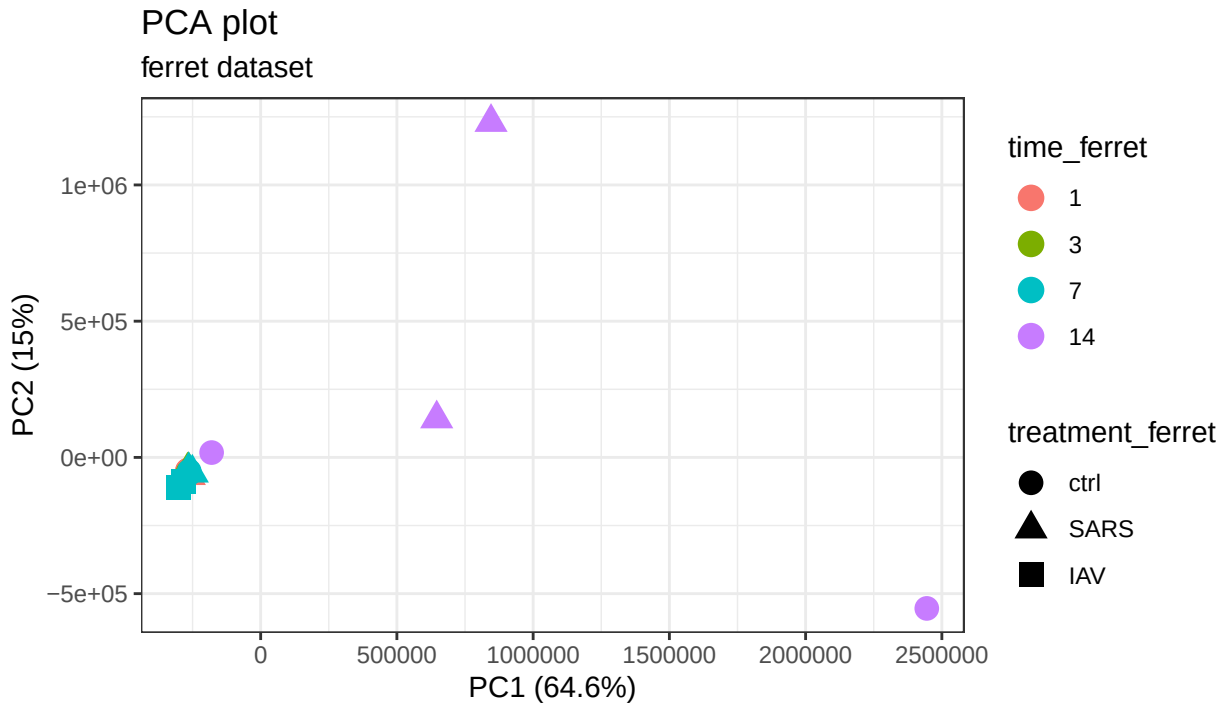
Filtering was carried out to remove lowly expressed genes. The threshold count is somehow higher than for human data, and does not filter all the lowly expressed genes, but looking at the non-normalized data it seems important not to increase the threshold in order to detetct relatively lowly expressed genes in some groups. Genes with less than 1 count per million (CPM) in at least 5 or more samples filtered out. This reduced the number of genes from 32057 to 11771.

PCA plot ferret data

Time to see if infection with SARS-CoV-2 induces a particular gene expression signature, and if this one is different after infection with IAV. The treatment groups have been already factorized, see: ctrl, ctrl, SARS, SARS, ctrl, ctrl, SARS, SARS, ctrl, ctrl, SARS, SARS, IAV, IAV, ctrl, ctrl, SARS, SARS. The idea here is to look at the PCA and create module for clust.

```
ferret_pca_res <- prcomp(t(data.matrix(ferret_DGEList_filtered_norm)), scale.=F, retx=T)
ferret_pc_var <- ferret_pca_res$sdev^2
ferret_pc_per <- round(ferret_pc_var/sum(ferret_pc_var) * 100, 1)
ferret_pca_res_df <- as_tibble(ferret_pca_res$x)
```

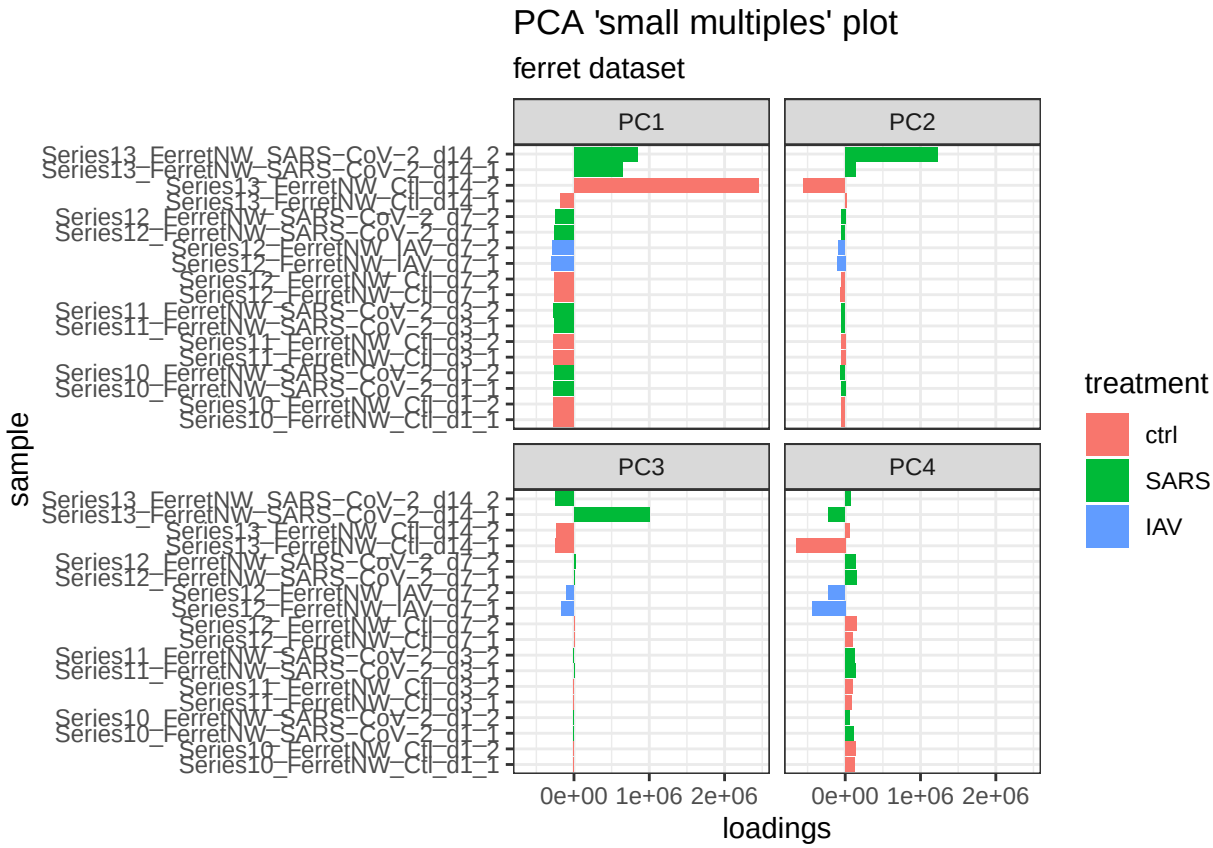
```
ggplot(ferret_pca_res_df) +
  aes(x = PC1, y = PC2, label = ferret_sample_labels,
      color = time_ferret,
      shape = treatment_ferret) +
  geom_point(size = 4) +
  #stat_ellipse() +
  xlab(paste0("PC1 (", ferret_pc_per[1], "%", ")")) +
  ylab(paste0("PC2 (", ferret_pc_per[2], "%", ")")) +
  labs(title = "PCA plot",
       subtitle = "ferret dataset") +
  coord_fixed() +
  theme_bw()
```



This is not easy to interpret although PC1 and PC2 account for 79.6% of the total variance. It seems that no difference is detected at earlier time points but only between control and SARS groups at day 14 post-infection, though duplicates are spread. Let's have a look at other PCs.

```
pca.res.df <- ferret_pca_res$x[, 1:4] %>%
  as_tibble() %>%
  add_column(sample = ferret_sample_labels,
             treatment = treatment_ferret)

pca.pivot <- pivot_longer(pca.res.df,
                          cols = PC1:PC4,
                          names_to = "PC",
                          values_to = "loadings")
```

```
ggplot(pca.pivot) +
  aes(x = sample, y = loadings, fill = treatment) +
  geom_bar(stat = "identity") +
  facet_wrap(~PC) +
  labs(title = "PCA 'small multiples' plot",
       subtitle = "ferret dataset") +
  theme_bw() +
  coord_flip()
```

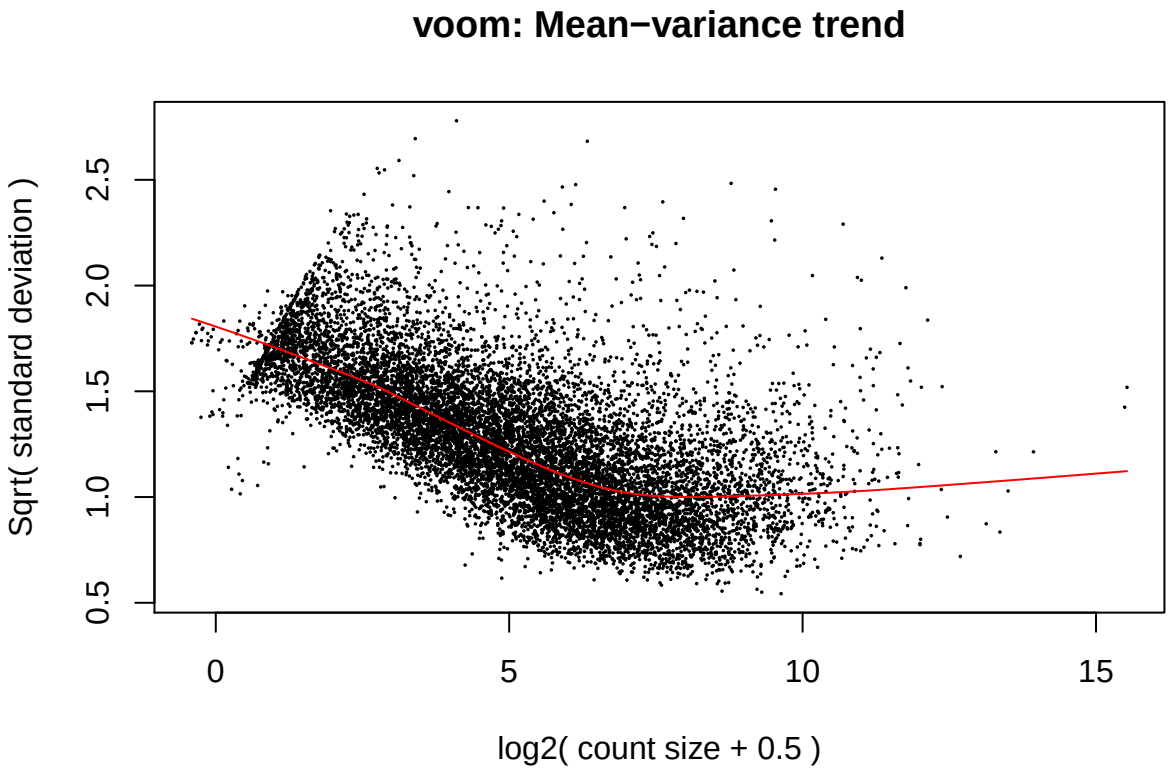


That is really weird, one sample from the control group at day 14 seems to belong to the SARS group.

Differentially expressed genes ferret

Let's see if there is any interesting gene differentially expressed detected between SARS-CoV-2 and mock conditions.

```
library(gt)

design <- model.matrix(~0 + treatment_ferret)
colnames(design) <- levels(treatment_ferret)

v_ferret_DGEList_filtered_norm <- voom(ferret_DGEList_filtered_norm, design, plot = T)
```

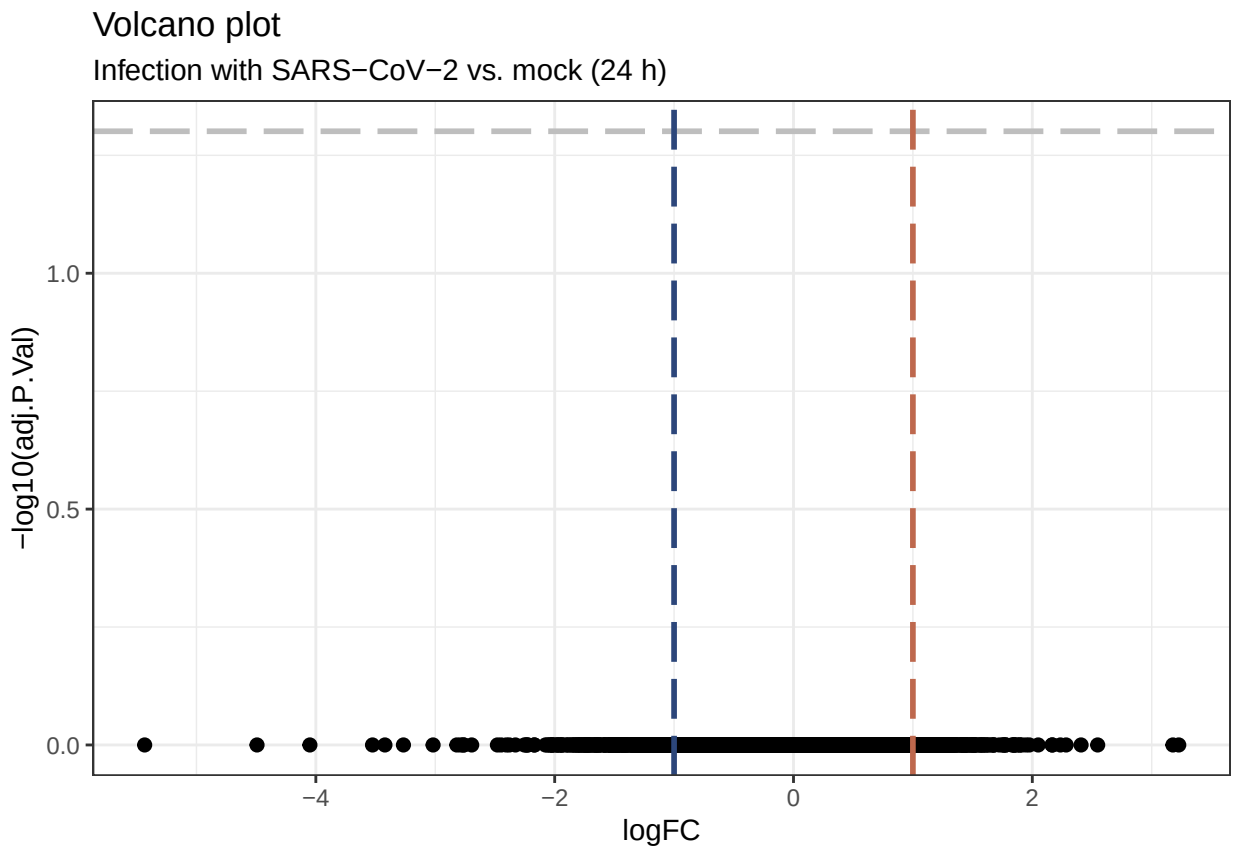



```
ferret_fit <- lmFit(v_ferret_DGEList_filtered_norm, design)
ferret_contrast_matrix <- makeContrasts(infection = SARS - ctrl,
                                        levels = design)

ferret_fits <- contrasts.fit(ferret_fit, ferret_contrast_matrix)
ferret_ebFit <- eBayes(ferret_fits)

ferret_results <- decideTests(ferret_ebFit,
                              method = "global",
                              adjust.method = "BH",
                              p.value = 0.01,
                              lfc = 1)
colnames(v_ferret_DGEList_filtered_norm$E) <- ferret_sample_labels
ferret_diffGenes <- v_ferret_DGEList_filtered_norm$E[ferret_results[, 1] !=0, ]
ferret_diffGenes_df <- as_tibble(ferret_diffGenes, rownames = "geneID")

#ferret_diffGenes_df %>%
#  sample_n(10) %>%
#  gt() %>%
#  fmt_number(columns = 2:11, decimals = 2)
```

The list is empty, let's check on a volcano plot.

```
ferret_top_hits <- topTable(ferret_ebFit,
                            adjust = "BH",
                            coef = 1,
                            number = 20000,
```

```
                            sort.by = "logFC")

ferret_top_hits_df <- ferret_top_hits %>%
  as_tibble(rownames = "geneID")

ggplot(ferret_top_hits_df) +
  aes(y = -log10(adj.P.Val), x = logFC, text = paste("Symbol:", geneID)) +
  geom_point(size = 2) +
  geom_hline(yintercept = -log10(0.05), linetype="longdash", colour="grey", size=1) +
  geom_vline(xintercept = 1, linetype="longdash", colour="#BE684D", size=1) +
  geom_vline(xintercept = -1, linetype="longdash", colour="#2C467A", size=1) +
  labs(title="Volcano plot",
       subtitle = "Infection with SARS-CoV-2 vs. mock (24 h)") +
  theme_bw()
```

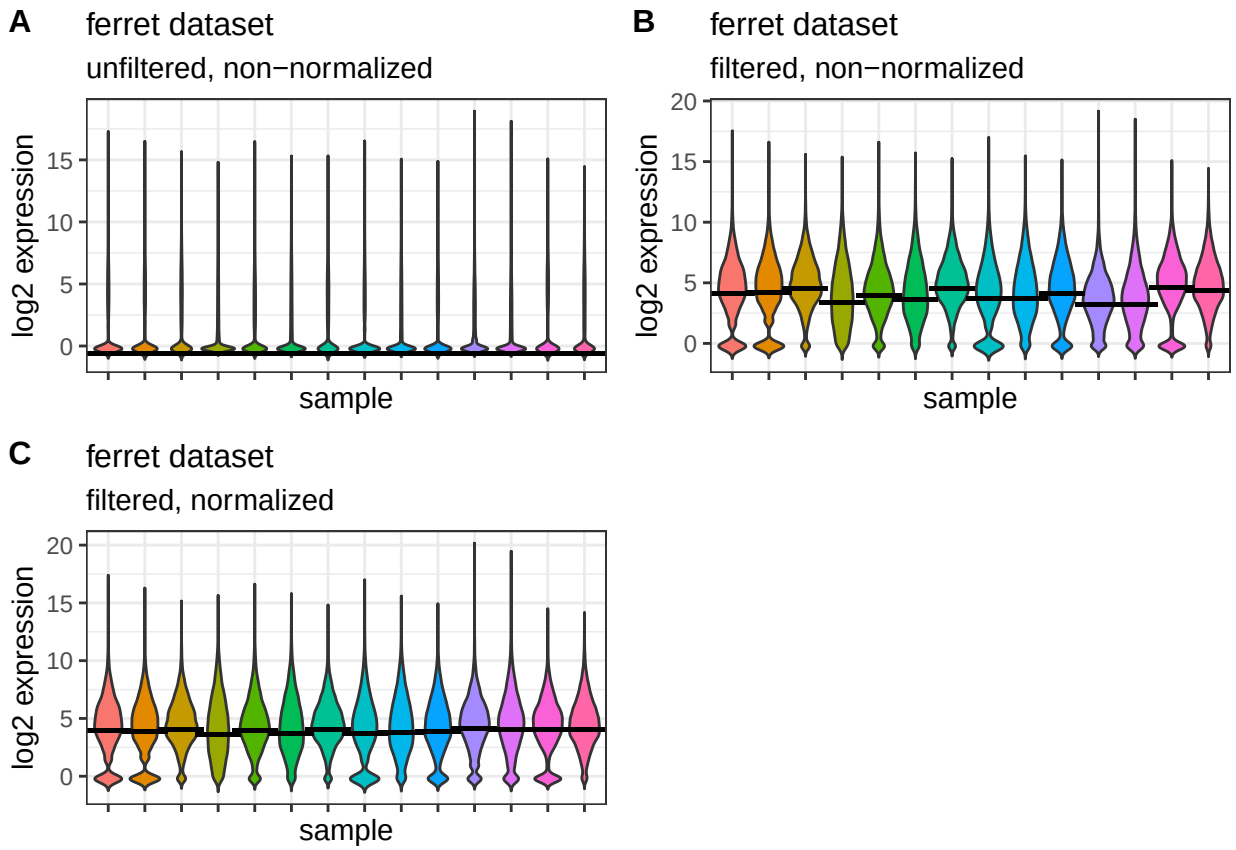


The analysis looks compromised for this timepoint as not significant logFC are detected, and we won't produce a clust. Let's filter this latest time point and redo all analyses for the ferret dataset.

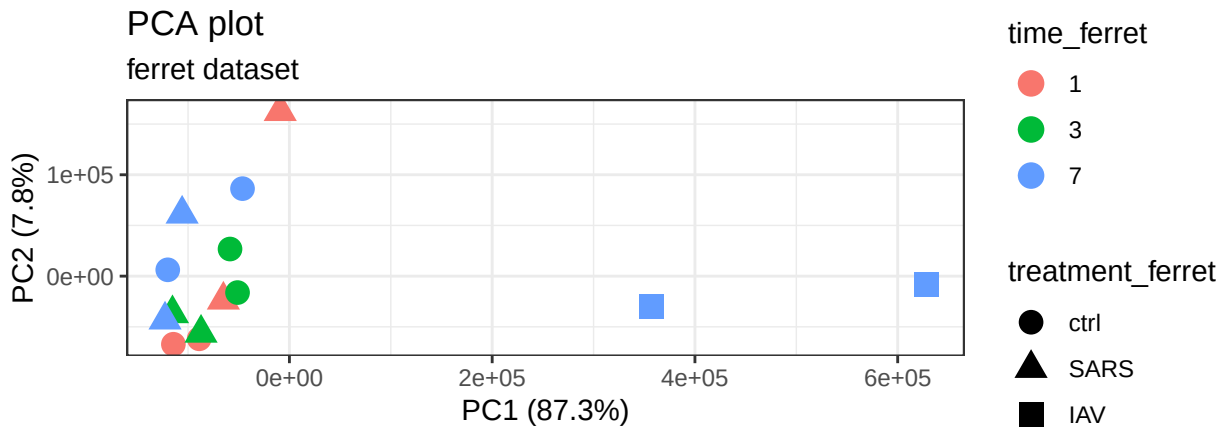
```
# we first select the columns of interest and further filter of ferret data
targets_ferret <- targets %>%
  dplyr::select(sample, host_common_name, tissue_cell_type, treatment, timepoint_postTreament) %>% # be
  dplyr::filter(host_common_name == "ferret" & tissue_cell_type == "Nasal Wash" & timepoint_postTreamen

# then we don't need the host_common_name and tissue_cell_type anymore
targets_ferret <- dplyr::select(targets_ferret, sample, treatment, timepoint_postTreament)
colnames(targets_ferret) <- c("sample", "treatment", "time")
```

```r
treatment_ferret <- factor(targets_ferret$treatment,
                           levels = c("Mock treatment", "SARS-CoV-2 infected (5E4 PFU)",
                                      "IAV infected (1E5 PFU)"),
                           labels = c("ctrl", "SARS", "IAV"))
time_ferret <- factor(targets_ferret$time,
                      levels = c("day1", "day3", "day7"),
                      labels = c(1, 3, 7))

# and finally we keep only the columns of interest in the count matrix
ferret_data <- ferret_covid_data[, targets_ferret$sample]
```

```r
ferret_sample_labels <- targets_ferret$sample
ferret_DGEList <- DGEList(ferret_data)
# we take advantage of the profile_function that return violin_plot of cpm
ferret_p1 <- profile_function(data = ferret_DGEList,
                              samples = ferret_sample_labels,
                              title = "ferret dataset",
                              subtitle = "unfiltered, non-normalized")

ferret_cpm <- cpm(ferret_DGEList)
ferret_keepers <- rowSums(ferret_cpm > 10) >= 2 # all duplicates
ferret_DGEList_filtered <- ferret_DGEList[ferret_keepers, ]
ferret_p2 <- profile_function(data = ferret_DGEList_filtered,
                              samples = ferret_sample_labels,
                              title = "ferret dataset",
                              subtitle = "filtered, non-normalized")

ferret_DGEList_filtered_norm <- calcNormFactors(ferret_DGEList_filtered,
                                                method = "TMM")
ferret_p3 <- profile_function(data = ferret_DGEList_filtered_norm,
                              samples = ferret_sample_labels,
                              title = "ferret dataset",
                              subtitle = "filtered, normalized")

plot_grid(ferret_p1, ferret_p2, ferret_p3,
          labels = c('A', 'B', 'C'),
          label_size = 12)
```

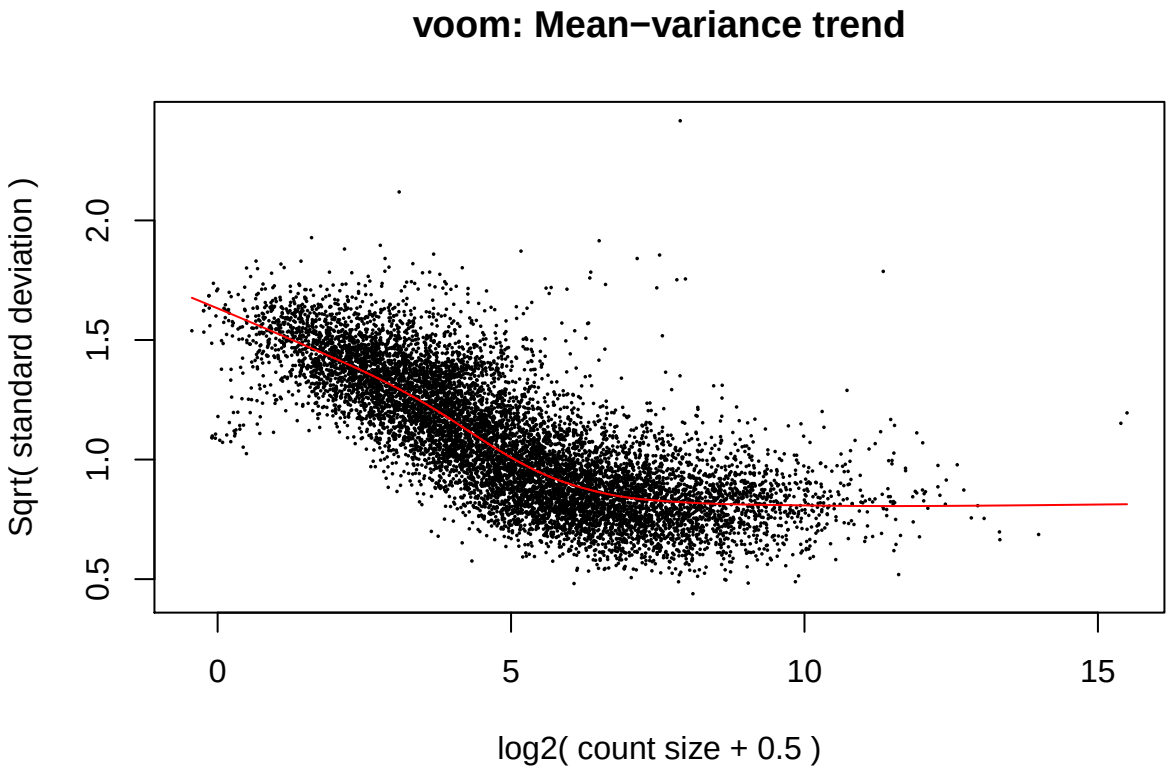



```
ferret_pca_res <- prcomp(t(data.matrix(ferret_DGEList_filtered_norm)), scale.=F, retx=T)
ferret_pc_var <- ferret_pca_res$sdev^2
ferret_pc_per <- round(ferret_pc_var/sum(ferret_pc_var) * 100, 1)
ferret_pca_res_df <- as_tibble(ferret_pca_res$x)

ggplot(ferret_pca_res_df) +
  aes(x = PC1, y = PC2, label = ferret_sample_labels,
      color = time_ferret,
      shape = treatment_ferret) +
  geom_point(size = 4) +
  #stat_ellipse() +
  xlab(paste0("PC1 (", ferret_pc_per[1], "%", ")")) +
  ylab(paste0("PC2 (", ferret_pc_per[2], "%", ")")) +
  labs(title = "PCA plot",
       subtitle = "ferret dataset") +
  coord_fixed() +
  theme_bw()
```



Now it looks better, and while PC1 and PC2 account for 95.1% of the total variance, we can see IAV infection induces a specific gene expression signature compared to SARS at day 7 post-infection. Interestingly, SARS differs to some extent to the control group at day 3.

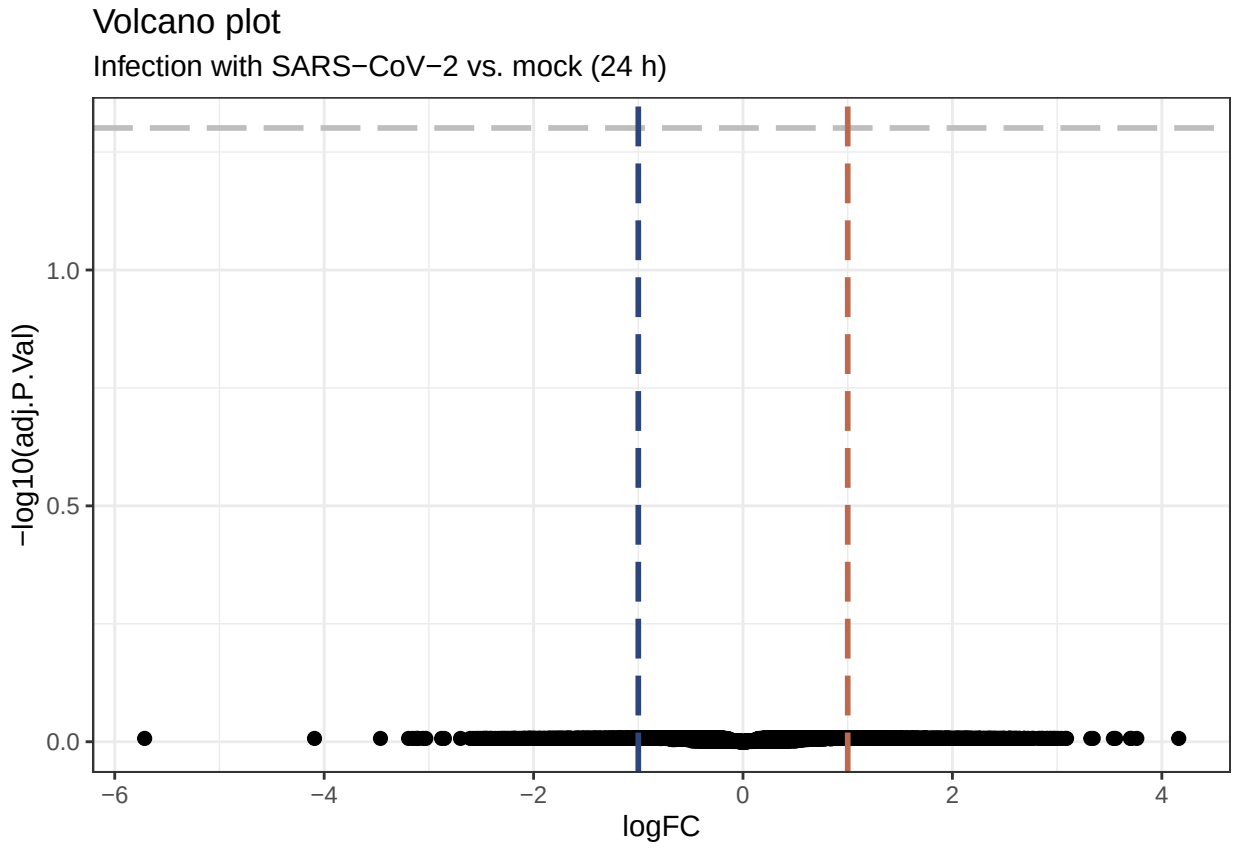
```
library(gt)

design <- model.matrix(~0 + treatment_ferret)
colnames(design) <- levels(treatment_ferret)

v_ferret_DGEList_filtered_norm <- voom(ferret_DGEList_filtered_norm, design, plot = T)
```




```
ferret_fit <- lmFit(v_ferret_DGEList_filtered_norm, design)
ferret_contrast_matrix <- makeContrasts(infection = SARS - ctrl,
                                        levels = design)

ferret_fits <- contrasts.fit(ferret_fit, ferret_contrast_matrix)
ferret_ebFit <- eBayes(ferret_fits)

ferret_results <- decideTests(ferret_ebFit,
                              method = "global",
                              adjust.method = "BH",
                              p.value = 0.01,
                              lfc = 1)
colnames(v_ferret_DGEList_filtered_norm$E) <- ferret_sample_labels
ferret_diffGenes <- v_ferret_DGEList_filtered_norm$E[ferret_results[, 1] !=0, ]
ferret_diffGenes_df <- as_tibble(ferret_diffGenes, rownames = "geneID")
```

```
ferret_top_hits <- topTable(ferret_ebFit,
                            adjust = "BH",
                            coef = 1,
                            number = 20000,
                            sort.by = "logFC")

ferret_top_hits_df <- ferret_top_hits %>%
  as_tibble(rownames = "geneID")

ggplot(ferret_top_hits_df) +
```

```
aes(y = -log10(adj.P.Val), x = logFC, text = paste("Symbol:", geneID)) +
geom_point(size = 2) +
geom_hline(yintercept = -log10(0.05), linetype="longdash", colour="grey", size=1) +
geom_vline(xintercept = 1, linetype="longdash", colour="#BE684D", size=1) +
geom_vline(xintercept = -1, linetype="longdash", colour="#2C467A", size=1) +
labs(title="Volcano plot",
     subtitle = "Infection with SARS-CoV-2 vs. mock (24 h)") +
theme_bw()
```



Again no significantly different gene expression could be observe after infection with SARS.

Conclusions

The data are not super convincing to me, no huge gene signature popped up upon infection with SARS-CoV-2. Moreover there is not really a good rationale to compare data from IAV group as the infection time are different. Last point was to detect modules from *in vivo* experiment with ferret using the clust software. Because de differential expression gene analysis was not conclusive, we did not run clust.

Altogether, neither the *ex vivo* experiment not the *in vivo* data generated in ferret go in the direction of a dramatic gene signature in response to SARS-CoV-2 infection.

Session info

The output from running 'sessionInfo' is shown below and details all packages and version necessary to reproduce the results in this report.

```
sessionInfo()
```

```
## R version 4.0.0 (2020-04-24)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19041)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=French_France.1252  LC_CTYPE=French_France.1252
## [3] LC_MONETARY=French_France.1252 LC_NUMERIC=C
## [5] LC_TIME=French_France.1252
##
## attached base packages:
## [1] stats4    parallel  stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] enrichplot_1.8.1       msigdbr_7.1.1          clusterProfiler_3.16.0
##  [4] gprofiler2_0.1.9       GSVA_1.36.0            GSEABase_1.50.0
##  [7] graph_1.66.0           annotate_1.66.0        XML_3.99-0.3
## [10] AnnotationDbi_1.50.0   IRanges_2.22.1         S4Vectors_0.26.1
## [13] Biobase_2.48.0         BiocGenerics_0.34.0    RColorBrewer_1.1-2
## [16] gplots_3.0.3           gt_0.2.1               cowplot_1.0.0
## [19] matrixStats_0.56.0     edgeR_3.30.0           limma_3.44.1
## [22] forcats_0.5.0          stringr_1.4.0          dplyr_0.8.5
## [25] purrr_0.3.4            readr_1.3.1            tidyr_1.0.3
## [28] tibble_3.0.1           ggplot2_3.3.0          tidyverse_1.3.0
## [31] knitr_1.28             tinytex_0.23           rmarkdown_2.2
##
## loaded via a namespace (and not attached):
##   [1] snow_0.4-3                readxl_1.3.1
##   [3] backports_1.1.7           fastmatch_1.1-0
##   [5] plyr_1.8.6                igraph_1.2.5
##   [7] lazyeval_0.2.2            splines_4.0.0
##   [9] BiocParallel_1.22.0       GenomeInfoDb_1.24.0
##  [11] urltools_1.7.3            digest_0.6.25
##  [13] htmltools_0.4.0           GOSemSim_2.14.0
##  [15] viridis_0.5.1             GO.db_3.11.1
##  [17] gdata_2.18.0              fansi_0.4.1
##  [19] magrittr_1.5              checkmate_2.0.0
##  [21] memoise_1.1.0             graphlayouts_0.7.0
##  [23] modelr_0.1.8              prettyunits_1.1.1
##  [25] colorspace_1.4-2          ggrepel_0.8.2
##  [27] blob_1.2.1                rvest_0.3.5
##  [29] haven_2.2.0               xfun_0.14
##  [31] crayon_1.3.4              RCurl_1.98-1.2
##  [33] jsonlite_1.6.1            scatterpie_0.1.4
##  [35] glue_1.4.1                polyclip_1.10-0
##  [37] gtable_0.3.0              zlibbioc_1.34.0
##  [39] XVector_0.28.0            DelayedArray_0.14.0
##  [41] scales_1.1.1              DOSE_3.14.0
##  [43] DBI_1.1.0                 Rcpp_1.0.4.6
##  [45] progress_1.2.2            viridisLite_0.3.0
```

```
##  [47] xtable_1.8-4               gridGraphics_0.5-0
##  [49] europepmc_0.3              bit_1.1-15.2
##  [51] htmlwidgets_1.5.1          httr_1.4.1
##  [53] fgsea_1.14.0               ellipsis_0.3.1
##  [55] pkgconfig_2.0.3            farver_2.0.3
##  [57] dbplyr_1.4.3               locfit_1.5-9.4
##  [59] ggplotify_0.0.5            tidyselect_1.1.0
##  [61] labeling_0.3               rlang_0.4.6
##  [63] reshape2_1.4.4             later_1.0.0
##  [65] munsell_0.5.0              cellranger_1.1.0
##  [67] tools_4.0.0                downloader_0.4
##  [69] cli_2.0.2                  generics_0.0.2
##  [71] RSQLite_2.2.0              ggridges_0.5.2
##  [73] broom_0.5.6                evaluate_0.14.1
##  [75] fastmap_1.0.1              yaml_2.2.1
##  [77] bit64_0.9-8                fs_1.4.1
##  [79] tidygraph_1.2.0            caTools_1.18.0
##  [81] ggraph_2.0.3               nlme_3.1-147
##  [83] mime_0.9.1                 DO.db_2.9
##  [85] xml2_1.3.2                 compiler_4.0.0
##  [87] shinythemes_1.1.2          rstudioapi_0.11
##  [89] plotly_4.9.2.1             reprex_0.3.0
##  [91] tweenr_1.0.1               stringi_1.4.6
##  [93] lattice_0.20-41            Matrix_1.3-0
##  [95] vctrs_0.3.0                pillar_1.4.4
##  [97] lifecycle_0.2.0            BiocManager_1.30.10
##  [99] triebeard_0.3.0            data.table_1.12.8
## [101] bitops_1.0-6               httpuv_1.5.2
## [103] GenomicRanges_1.40.0       qvalue_2.20.0
## [105] R6_2.4.1                   promises_1.1.0
## [107] KernSmooth_2.23-17         gridExtra_2.3
## [109] MASS_7.3-51.6              gtools_3.8.2
## [111] assertthat_0.2.1           SummarizedExperiment_1.18.1
## [113] withr_2.2.0                GenomeInfoDbData_1.2.3
## [115] hms_0.5.3                  grid_4.0.0
## [117] rvcheck_0.1.8              ggforce_0.3.1
## [119] shiny_1.4.0.2              lubridate_1.7.8
```