RNA-Seq Exploratory Analysis

2024-08-29

```
suppressPackageStartupMessages({
    library(here)
    library(tidyverse)
    library(biomaRt)
    library(factoextra)
    library(ggrepel)
    library(caret)

    library(DESeq2)
    library(enrichR)
    library(xlsx)
    library(circlize)
    library(ComplexHeatmap)
})

main_folder <- here()
```

```
count_paths <- list.files(path = file.path(main_folder,
    "count_data"), pattern = "quant.genes.sf",
    recursive = T, full.names = T)
count_lists <- lapply(count_paths, function(x) read_tsv(x,
    col_types = cols()))
names <- lapply(basename(count_paths), function(x) strsplit(x,
    split = "-")[[1]][1])
names(count_lists) <- names
counts <- count_lists %>%
    lapply(., function(df) {
        df %>%
            dplyr::select(Name, NumReads)
    }) %>%
    # Map(function(x, n) setNames(x,
    # c(names(x)[1], n)), ., names(.))
Map(function(x, n) setNames(x, c("Ensembl_ID",
    n)), ., names(.)) %>%
    purrr::reduce(inner_join, by = c("Ensembl_ID")) %>%
    as.data.frame(.) %>%
    mutate_if(is.numeric, round)
# write.table(counts,
# file.path(main_folder,'gene_counts.tsv'),
# quote = F, sep = '\t', row.names = F,
# col.names = T)
```

```
tpm <- count_lists %>%
    lapply(., function(df) {
        df %>%
            dplyr::select(Name, TPM)
    }) %>%
    Map(function(x, n) setNames(x, c("Ensembl_ID",
        n)), ., names(.)) %>%
    purrr::reduce(inner_join, by = c("Ensembl_ID")) %>%
    as.data.frame(.)
```

Convert Ensembl IDs to gene names

```
ensembl <- useEnsembl(biomart = "genes",
    dataset = "hsapiens_gene_ensembl")
genes <- data.frame(ensembl = tpm$Ensembl_ID)
# run the query
gene_IDs <- getBM(attributes = c("ensembl_gene_id_version",
    "hgnc_symbol", "gene_biotype"), filters = c("ensembl_gene_id_version"),
    values = genes$ensembl, mart = ensembl)
head(gene_IDs)
```

```
##   ensembl_gene_id_version hgnc_symbol   gene_biotype
## 1      ENSG00000000457.14       SCYL3 protein_coding
## 2      ENSG00000000460.17       FIRRM protein_coding
## 3      ENSG00000000938.13         FGR protein_coding
## 4      ENSG00000000971.17         CFH protein_coding
## 5      ENSG00000001460.18       STPG1 protein_coding
## 6      ENSG00000001461.17      NIPAL3 protein_coding
```

tpm matrix is updated by matching to the retrieved annotation

```
gene_IDs <- gene_IDs %>%
    filter(!hgnc_symbol == "")

tpm <- tpm %>%
    dplyr::rename(ensembl_gene_id_version = "Ensembl_ID") %>%
    right_join(., gene_IDs) %>%
    dplyr::select(-gene_biotype, -ensembl_gene_id_version) %>%
    group_by(hgnc_symbol) %>%
    summarise_all(mean) %>%
    column_to_rownames("hgnc_symbol") %>%
    rownames_to_column("gene_id")
```

```
## Joining with `by = join_by(ensembl_gene_id_version)`
```

```
head(tpm)
```

```
##     gene_id 10067211 10067219 10067220 10067221 10067222 10067223 10067224
## 1      A1BG    0.000    0.051    0.300   23.762    0.000    0.000    0.035
## 2  A1BG-AS1    0.471    0.784    0.929    0.251    0.229    0.282    2.075
## 3       A2M   31.203    6.220   67.024  405.047   45.597    0.517   32.293
```

```
## 4   A2M-AS1    0.107    0.031    0.636    0.764    0.259    0.241    1.015
## 5     A2ML1    1.392   12.098   27.463    0.295    0.489    5.873    1.010
## 6 A2ML1-AS1    0.000    0.000    0.000    0.000    0.000    0.000    0.275
##   10067225 10067226 10067227 10067228 10067229 10067231 10067233 10067235
## 1    0.009    0.000    0.052    0.031    0.063    5.252    0.106    0.014
## 2    0.296    0.138    0.414    0.194    0.234    0.079    0.603    0.584
## 3   41.735   16.630   38.728   41.527   24.947  316.493   10.872   63.976
## 4    0.180    0.213    0.323    0.522    0.260    0.509    0.071    0.200
## 5    0.293    8.003   65.214   22.106    2.679    4.249  187.241  126.497
## 6    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000
##   10067236 10067238 10067241 10067243 10067245 10067246 10067247 10067248
## 1    0.000    0.016    0.079    0.014    0.126    0.073    0.000    0.128
## 2    0.000    0.212    0.393    0.412    1.389    0.505    0.009    1.161
## 3   31.622  118.599  154.566   34.882  231.743   84.548    3.288    5.764
## 4    0.359    0.620    0.803    0.172    3.606    0.484    0.083    0.100
## 5  102.149   87.757    1.626   37.808    0.000    1.299    3.621   66.894
## 6    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000
##   10067251 10067261 10067268 10067270 10067273 10067275 10067276 10067488
## 1    0.126    2.672    0.167    0.000    0.062    0.119    0.076    0.000
## 2    0.631    0.204    2.109    0.029    0.128    0.602    0.859    0.254
## 3   20.791  129.922    3.570    3.028    6.225    7.966   59.667   25.522
## 4    0.204    0.077    0.156    0.116    0.073    0.639    0.499    0.084
## 5    0.774  115.632    0.039   20.094   20.266    0.299    0.201   57.463
## 6    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000
##   10067489 10067490 10067492 10067493 10067495 10067496 10067497 10067498
## 1    0.015    0.088    0.051    0.048    0.053    0.214    0.000    0.200
## 2    0.829    1.531    1.321    1.210    0.223    2.852    0.136    2.095
## 3   21.699   12.238   46.893   67.824   19.376  290.766    3.689   31.495
## 4    0.464    0.000    0.169    0.295    0.209    1.271    0.033    0.080
## 5    8.295  127.471   91.009    7.284   95.977    0.225    8.516   58.071
## 6    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000
##   10067501 10067502 10067515 10067516 10067517 10067826 10067827 10067829
## 1    0.014    0.000    0.055    0.013    0.036    0.020    0.000    0.034
## 2    0.052    0.025    1.795    0.736    2.559    0.732    0.391    1.435
## 3   21.080    1.324    8.287   57.370   26.336   58.414   16.430  131.753
## 4    0.245    0.024    0.044    0.276    0.208    0.080    0.094    0.336
## 5    6.002    0.000  163.712    0.104   11.507   55.920  132.299  104.455
## 6    0.000    0.000    0.000    0.000    0.000    0.000    0.000    0.000
##   10067830 10067836 10067839
## 1    0.000    0.074    0.174
## 2    0.073    0.545    1.647
## 3   22.202   90.873   80.412
## 4    0.197    0.173    0.270
## 5   16.401    2.626   10.317
## 6    0.000    0.050    0.000
```

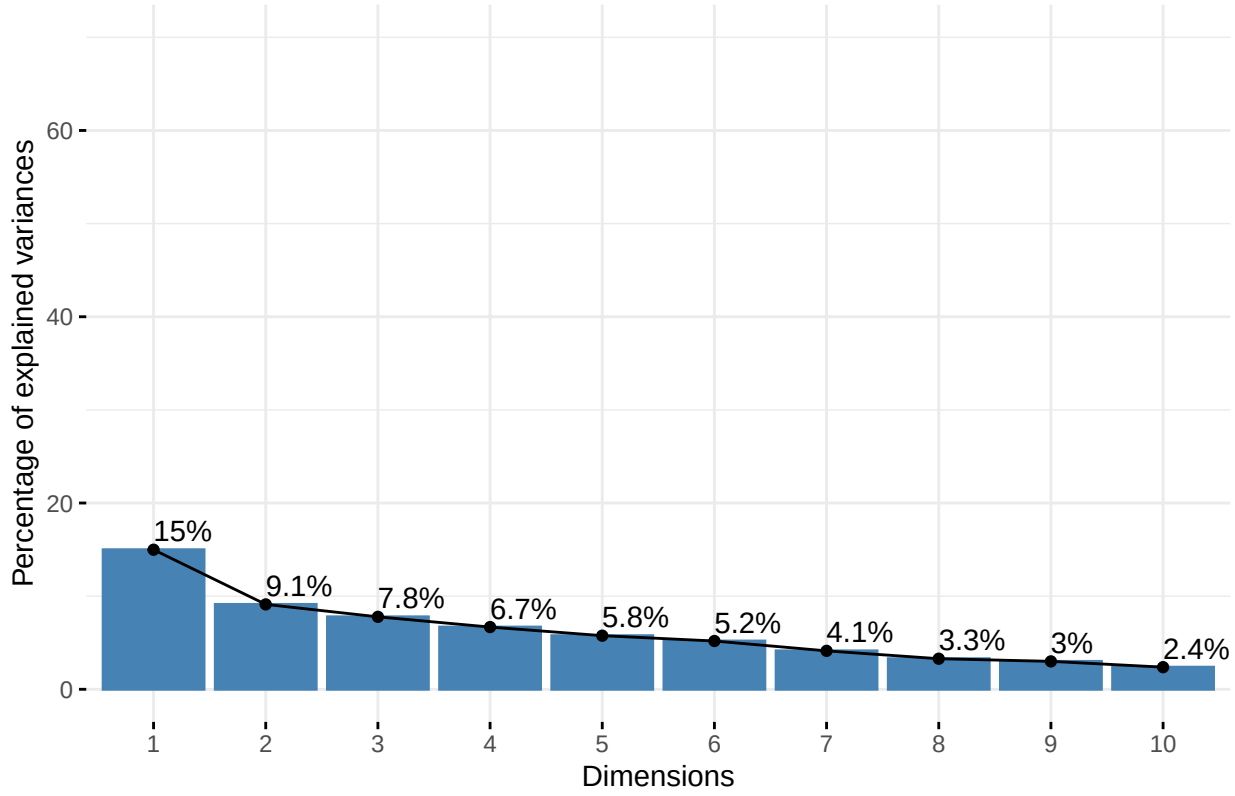
```
# write.table(tpm,
# file.path(main_folder,'acc_cibersort_input.tsv'),
# quote = F, sep = '\t', row.names = F,
# col.names = T)
```

```
vst <- read.table(file.path(main_folder,
    "differential_analysis_0824/deseq_vst_data.txt"))
colnames(vst) <- gsub("^X", "", colnames(vst))
```

```r
pca_data = prcomp(t(vst))
summary(pca_data)
```

```
## Importance of components:
##                             PC1      PC2      PC3      PC4      PC5      PC6
## Standard deviation     63.9867 49.90656 46.10558 42.72332 39.66570 37.63477
## Proportion of Variance  0.1499  0.09117  0.07781  0.06681  0.05759  0.05184
## Cumulative Proportion   0.1499  0.24103  0.31884  0.38566  0.44325  0.49509
##                             PC7      PC8      PC9     PC10     PC11     PC12
## Standard deviation     33.58370 29.95445 28.61948 25.48666 24.73158 23.52585
## Proportion of Variance  0.04128  0.03284  0.02998  0.02378  0.02239  0.02026
## Cumulative Proportion   0.53638  0.56922  0.59920  0.62298  0.64536  0.66562
##                            PC13     PC14     PC15     PC16     PC17     PC18
## Standard deviation     23.34444 22.62875 21.53305 20.93252 20.48017 20.02441
## Proportion of Variance  0.01995  0.01874  0.01697  0.01604  0.01535  0.01468
## Cumulative Proportion   0.68557  0.70431  0.72129  0.73733  0.75268  0.76736
##                            PC19     PC20    PC21    PC22     PC23     PC24
## Standard deviation     19.50283 18.97850 18.4789 18.2599 17.54030 17.24837
## Proportion of Variance  0.01392  0.01318  0.0125  0.0122  0.01126  0.01089
## Cumulative Proportion   0.78128  0.79446  0.8070  0.8192  0.83043  0.84132
##                            PC25     PC26     PC27     PC28     PC29     PC30
## Standard deviation     16.76886 16.48292 16.01702 15.96123 15.61564 15.40634
## Proportion of Variance  0.01029  0.00994  0.00939  0.00933  0.00893  0.00869
## Cumulative Proportion   0.85161  0.86155  0.87095  0.88027  0.88920  0.89788
##                            PC31     PC32     PC33     PC34    PC35     PC36
## Standard deviation     14.73666 14.48728 14.35203 13.90651 13.6250 13.31766
## Proportion of Variance  0.00795  0.00768  0.00754  0.00708  0.0068  0.00649
## Cumulative Proportion   0.90583  0.91352  0.92106  0.92813  0.9349  0.94142
##                            PC37     PC38     PC39     PC40     PC41     PC42
## Standard deviation     13.00365 12.54243 12.41856 12.28849 12.11249 11.86366
## Proportion of Variance  0.00619  0.00576  0.00565  0.00553  0.00537  0.00515
## Cumulative Proportion   0.94761  0.95337  0.95901  0.96454  0.96991  0.97506
##                            PC43     PC44     PC45     PC46    PC47    PC48
## Standard deviation     11.26985 10.67636 10.36469 10.20151 9.08529 8.84174
## Proportion of Variance  0.00465  0.00417  0.00393  0.00381 0.00302 0.00286
## Cumulative Proportion   0.97971  0.98389  0.98782  0.99163 0.99465 0.99751
##                            PC49      PC50
## Standard deviation     8.24825 1.272e-13
## Proportion of Variance 0.00249 0.000e+00
## Cumulative Proportion  1.00000 1.000e+00
```

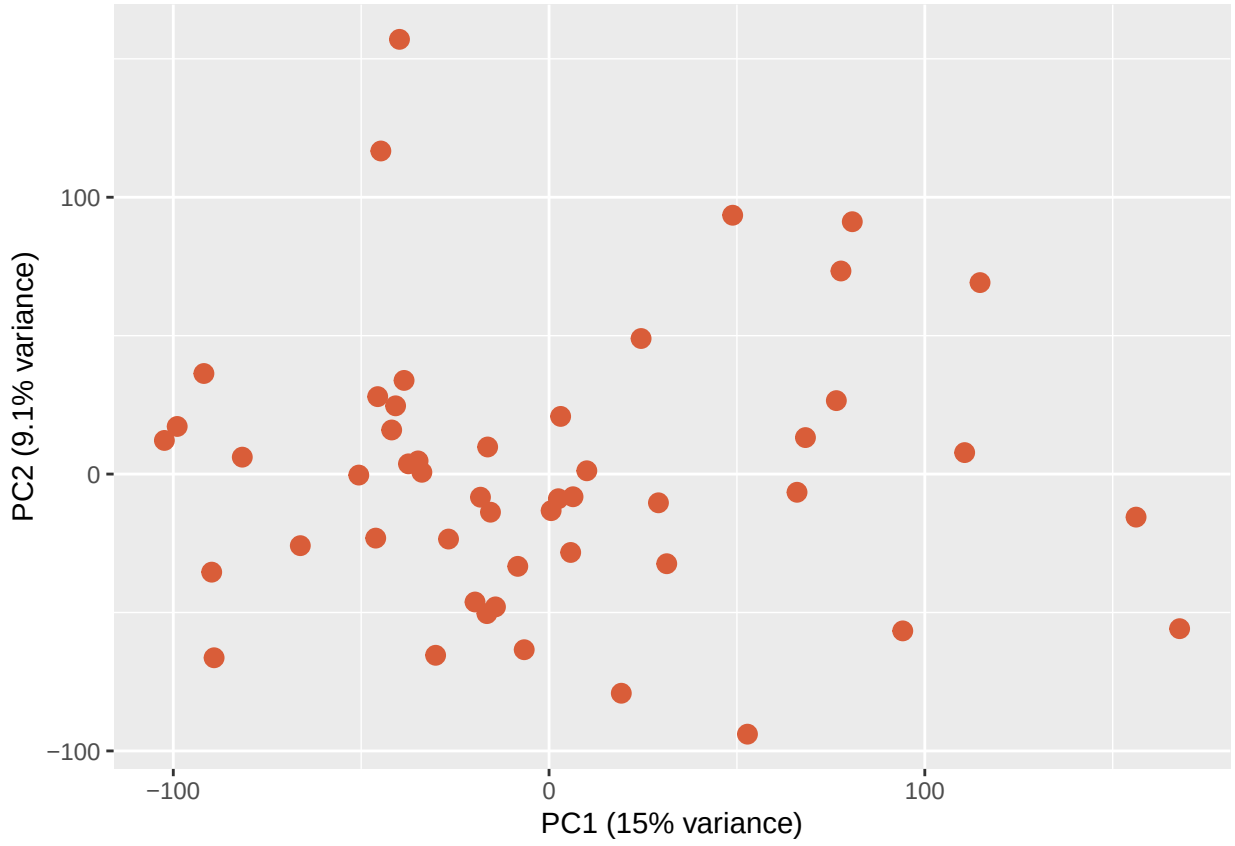
```r
p1 <- fviz_eig(pca_data, addlabels = TRUE,
    ylim = c(0, 70), main = "")
p1
```



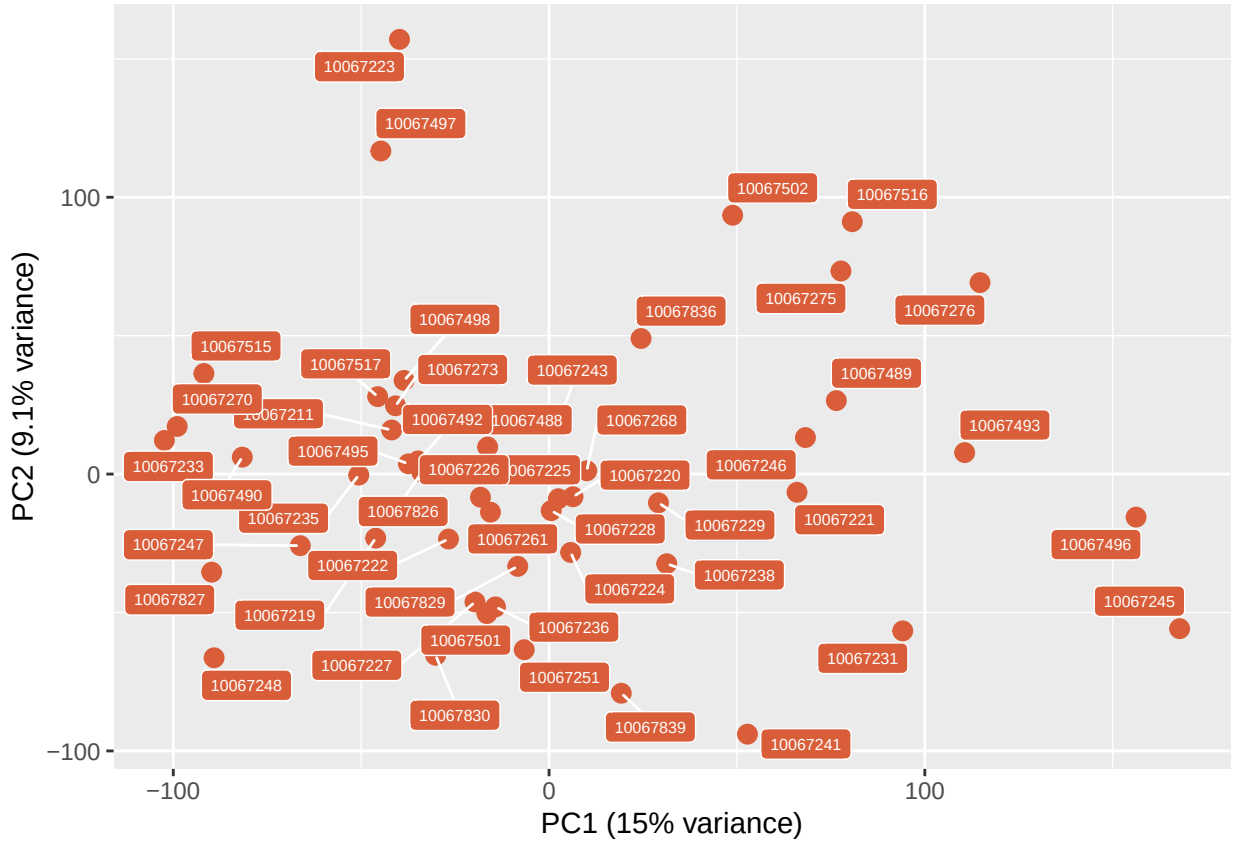

```
png(file.path(main_folder, "pca_screeplot.png"),
    res = 300, units = "px", width = 2500,
    height = 2000)
p1
dev.off()
```

```
## pdf
##   2
```

```
pca_data_perc = round(100 * pca_data$sdev^2/sum(pca_data$sdev^2),
    1)
df_pca_data = data.frame(PC1 = pca_data$x[,
    1], PC2 = pca_data$x[, 2], sample = colnames(vst))

# PCA for components 1&2
p2 <- ggplot(df_pca_data, aes(PC1, PC2, colour = "#d95d39")) +
    geom_point(size = 3) + scale_color_manual(values = c("#d95d39")) +
    labs(x = paste0("PC1 (", pca_data_perc[1],
        "% variance)"), y = paste0("PC2 (",
        pca_data_perc[2], "% variance)")) +
    theme(legend.position = "none")
p2
```




```
# PCA with labels
p3 <- ggplot(df_pca_data, aes(PC1, PC2, colour = "#d95d39")) +
    geom_point(size = 3, show.legend = F) +
    scale_color_manual(values = c("#d95d39")) +
    geom_label_repel(aes(label = sample,
        fill = "#d95d39"), color = "white",
        size = 2, max.overlaps = Inf) + scale_fill_manual(values = c("#d95d39")) +
    labs(x = paste0("PC1 (", pca_data_perc[1],
        "% variance)"), y = paste0("PC2 (",
        pca_data_perc[2], "% variance)")) +
    theme(legend.position = "none")
p3
```




```
png(file.path(main_folder, "PC1_vs_PC2.png"),
    res = 300, units = "px", width = 2500,
    height = 2000)
p2
dev.off()
```

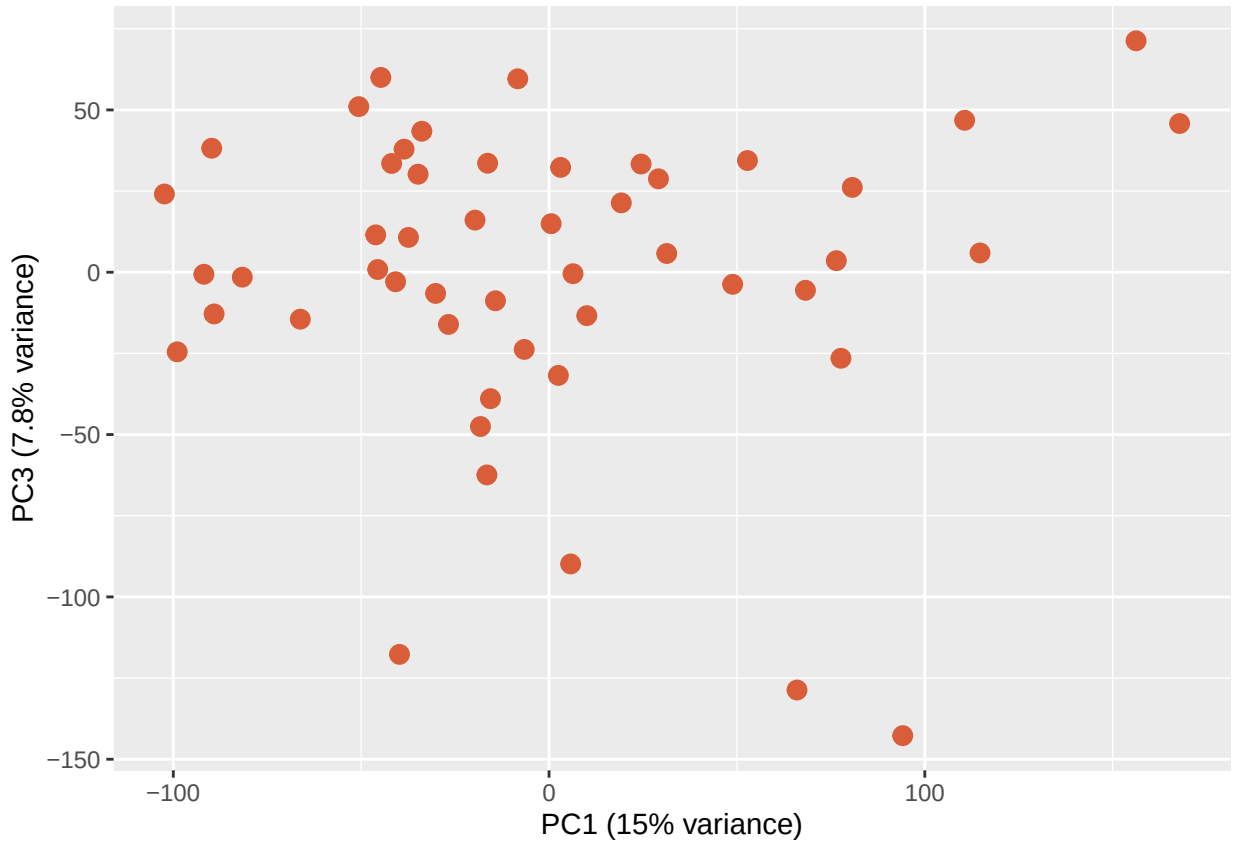
```
## pdf
##   2
```

```
png(file.path(main_folder, "PC1_vs_PC2_labels.png"),
    res = 300, units = "px", width = 2500,
    height = 2000)
p3
dev.off()
```

```
## pdf
##   2
```

```
# PCA for components 1&3
df_pca_data = data.frame(PC1 = pca_data$x[,
    1], PC3 = pca_data$x[, 3], sample = colnames(vst))

p4 <- ggplot(df_pca_data, aes(PC1, PC3, colour = "#d95d39")) +
    geom_point(size = 3) + scale_color_manual(values = c("#d95d39")) +
```

```
    labs(x = paste0("PC1 (", pca_data_perc[1],
        "% variance)"), y = paste0("PC3 (",
        pca_data_perc[3], "% variance)")) +
    theme(legend.position = "none")
p4
```




```
png(file.path(main_folder, "PC1_vs_PC3.png"),
    res = 300, units = "px", width = 2500,
    height = 2000)
p4
dev.off()
```

```
## pdf
##   2
```

```
metadata <- readxl::read_xlsx(file.path(main_folder,
    "Recap_PEVO_data_RNAseq.xlsx"))
samples <- data.frame(sample = colnames(counts)[2:51])

metadata_selection <- as.data.frame(inner_join(metadata,
    samples, by = join_by(`Tumor/RNA ID` ==
        sample)))
dim(metadata_selection)
```

```
## [1] 48  6
```

```
# missing samples
samples %>%
    filter(!sample %in% metadata_selection$`Tumor/RNA ID`)
```

```
##     sample
## 1 10067275
## 2 10067276
```

```
# Tumor/RNA ID entry in metadata file
# for those samples is 10067276/5
metadata_selection[nrow(metadata_selection) +
    1, ] = c("P-0022", "10067275", "10067150",
    "Cervix", "Cervix", "Baseline")
metadata_selection[nrow(metadata_selection) +
    1, ] = c("P-0022", "10067276", "10067150",
    "Cervix", "Cervix", "Baseline")
dim(metadata_selection)
```

```
## [1] 50  6
```

```
groups01 <- metadata_selection %>%
    dplyr::select(`Tumor/RNA ID`, `Timepoint of the tumor`) %>%
    dplyr::rename(sample = "Tumor/RNA ID",
        group = "Timepoint of the tumor")
groups01 <- groups01[match(colnames(counts)[2:51],
    groups01$sample), ]
groups01 %>%
    dplyr::count(group)
```

```
##         group  n
## 1    Baseline 38
## 2        C3D1  8
## 3 Progression  4
```

```
# baseline VS treatment with HDAC
# inhibitors C3D1, Day 1 of
# chemotherapy treatment cycle 3
comparisons01 <- data.frame(treatment = c("C3D1",
    "Progression"), control = c("Baseline",
    "C3D1"))
```

```
counts_hgnc <- counts %>%
    dplyr::rename(ensembl_gene_id_version = "Ensembl_ID") %>%
    right_join(., gene_IDs) %>%
    dplyr::select(-gene_biotype, -ensembl_gene_id_version) %>%
    group_by(hgnc_symbol) %>%
    summarise_all(mean) %>%
    column_to_rownames("hgnc_symbol") %>%
    rownames_to_column("gene_id")
```

```
## Joining with ‘by = join_by(ensembl_gene_id_version)‘
```

AUTOGO

```
files <- list.files(path = file.path(main_folder,
    "auto-go/R"), recursive = T, all.files = T,
    full.names = T)
invisible(sapply(files, source))

autogo_folder <- file.path(main_folder, "")
```

```
deseq_analysis(counts_hgnc, groups01, comparisons01,
    padj_threshold = 0.05, log2FC_threshold = 1,
    pre_filtering = T, save_excel = T, where_results = autogo_folder,
    outfolder = "differential_analysis_0824/")
```

```
## converting counts to integer mode

## estimating size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing

## -- replacing outliers and refitting for 2085 genes
## -- DESeq argument 'minReplicatesForReplace' = 7
## -- original counts are preserved in counts(dds)

## estimating dispersions

## fitting model and testing
```

TPM filtering on baseline expression

```
baseline <- groups01 %>%
    filter(group == "Baseline") %>%
    pull(sample)
c3d1 <- groups01 %>%
    filter(group == "C3D1") %>%
    pull(sample)

expression_filter_baseline <- tpm %>%
    dplyr::select(any_of(c("gene_id", baseline))) %>%
    column_to_rownames("gene_id")
expression_filter_baseline$mean_baseline <- apply(expression_filter_baseline,
    1, mean)
```

```
expression_filter_baseline <- expression_filter_baseline %>%
    rownames_to_column("gene_id")

expression_filter_c3d1 <- tpm %>%
    dplyr::select(any_of(c("gene_id", c3d1))) %>%
    column_to_rownames("gene_id")
expression_filter_c3d1$mean_c3d1 <- apply(expression_filter_c3d1,
    1, mean)
expression_filter_c3d1 <- expression_filter_c3d1 %>%
    rownames_to_column("gene_id")

expression_filter <- inner_join(expression_filter_baseline,
    expression_filter_c3d1, by = "gene_id") %>%
    dplyr::select(gene_id, mean_baseline,
        mean_c3d1)
filter_out <- expression_filter %>%
    filter(mean_baseline <= 1 | mean_c3d1 <=
        1)
```

```
path_res <- file.path(main_folder, "differential_analysis_0824/")
all_path_res <- list.files(path = path_res,
    pattern = "_allres.tsv", recursive = T,
    full.names = T)
res_lists <- lapply(all_path_res, function(x) read_tsv(x,
    col_types = cols()))
names(res_lists) <- gsub(paste0(path_res,
    "/|/DE_.*"), "", all_path_res)

# for (th in c(1)) {
# lapply(names(res_lists), function (i)
# volcanoplot(res_lists[[i]],
# my_comparison = i, log2FC_thresh =
# th, padj_thresh = 0.05, where_results
# = autogo_folder, outfolder =
# 'differential_analysis_0824/')) }

for (th in c(1)) {
    lapply(names(res_lists), function(i) volcanoplot_tpm(res_lists[[i]],
        my_comparison = i, log2FC_thresh = th,
        padj_thresh = 0.05, where_results = autogo_folder,
        outfolder = "differential_analysis_0824/"))
}
```

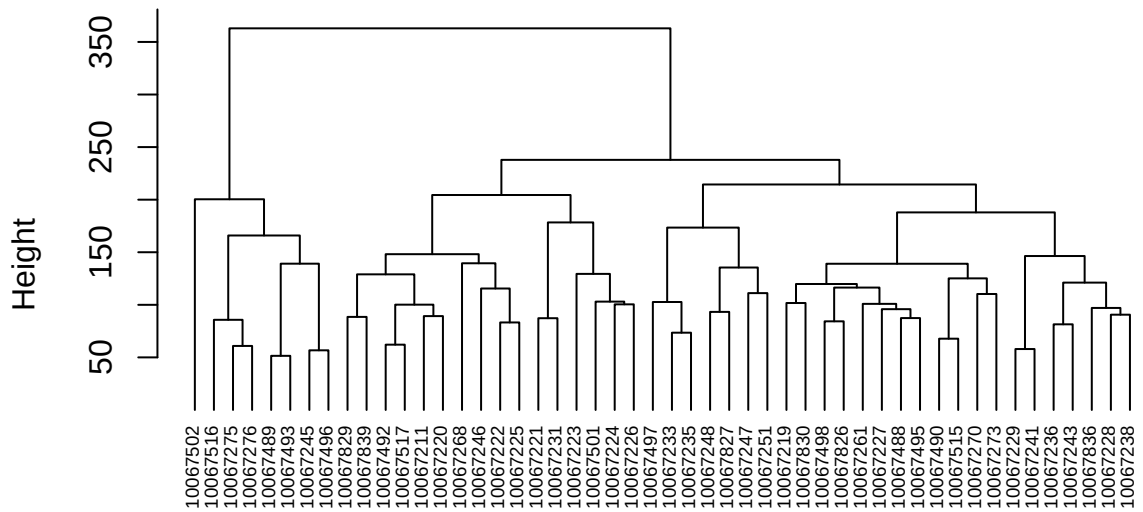
```
# Reading gene lists for enrichment
lista_p005_fc1 <- read_gene_list(where_results = autogo_folder,
    outfolder = "differential_analysis_0824/",
    log2FC_threshold = 1, padj_threshold = 0.05,
    which_list = "everything")

# Enrichment analysis
lapply(names(lista_p005_fc1), function(i) autoGO(lista_p005_fc1[[i]],
    my_comparison = i, dbs = c("GO_Molecular_Function_2021",
        "GO_Cellular_Component_2021", "GO_Biological_Process_2021",
```

```
        "KEGG_2021_Human"), where_results = autogo_folder,
    outfolder = "differential_analysis_0824/",
    excel = T))


tab_p005_fc1 <- read_enrich_tables(where_results = autogo_folder,
    outfolder = "differential_analysis_0824/",
    log2FC_threshold = 1, padj_threshold = 0.05,
    which_list = "everything")
invisible(lapply(names(tab_p005_fc1), function(i) barplotGO(tab_p005_fc1[[i]],
    my_comparison = i, where_results = autogo_folder,
    outfolder = "differential_analysis_0824/")))
invisible(lapply(names(tab_p005_fc1), function(i) lolliGO(tab_p005_fc1[[i]],
    my_comparison = i, where_results = autogo_folder,
    outfolder = "differential_analysis_0824/")))
```

```
groups02 <- metadata_selection %>%
    dplyr::select(`Tumor/RNA ID`, `Primary Cancer Type`) %>%
    dplyr::rename(sample = "Tumor/RNA ID",
        group = "Primary Cancer Type")
groups02 <- groups02[match(colnames(counts)[2:51],
    groups02$sample), ]
groups02 %>%
    dplyr::count(group)
```

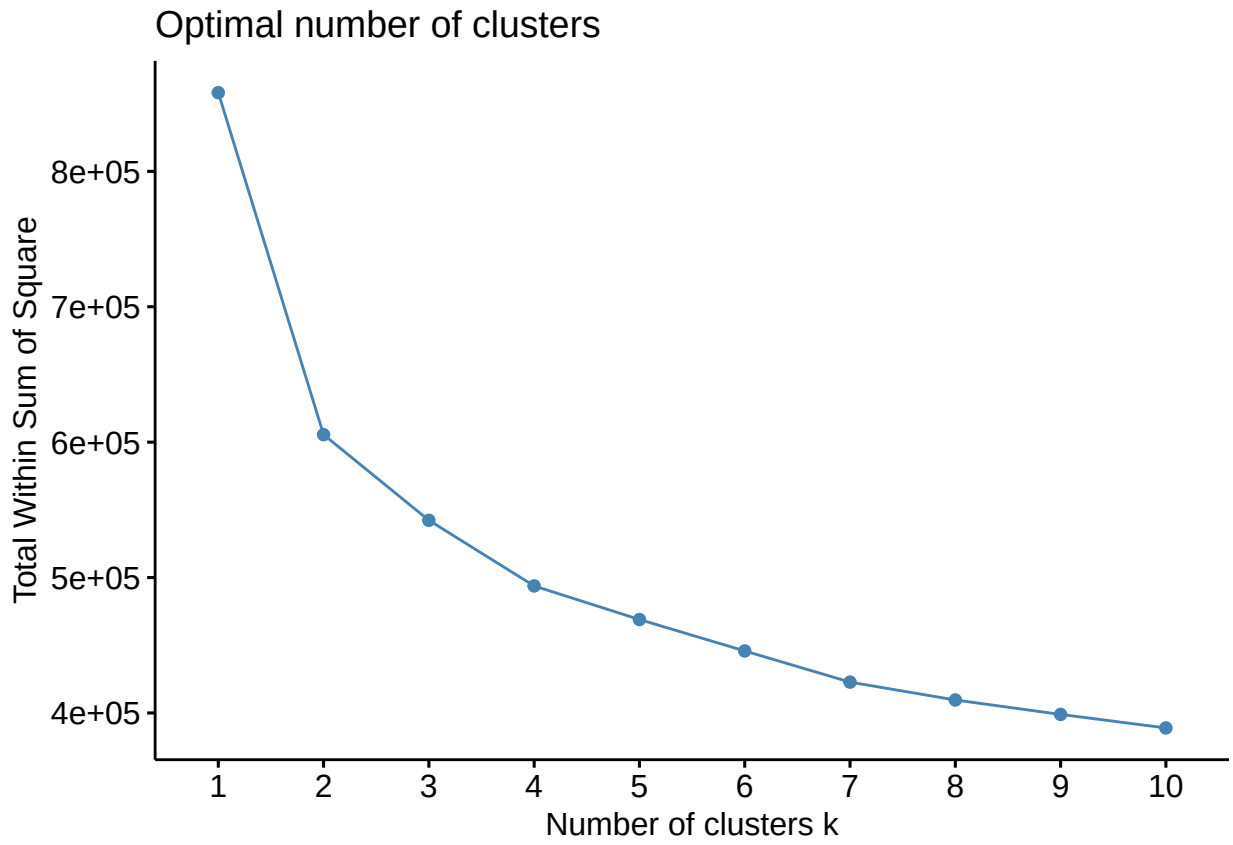
```
##           group  n
## 1          Anus 14
## 2        Cervix 15
## 3 Head and Neck  6
## 4          Lung  1
## 5         Penis  6
## 6         Vulva  8
```

```
## Hierarchical Clustering

# feature selection keep top genes
# based on median absolute deviation
mads <- apply(vst, 1, mad)
# selecting features
mad2k <- vst[rev(order(mads))[1:2000], ]

# calculate distances (default:
# Eucledian distance)
dist_matrix <- dist(t(mad2k))
# perform hierarchical clustering using
# ? linkage
hc <- hclust(dist_matrix, method = "ward.D2")
plot(hc, main = "", hang = -1, cex = 0.6,
    xlab = "")
```



hclust (*, "ward.D2")

```
fviz_nbclust(mad2k, FUN = hcut, method = "wss")
```




```
prediction <- cutree(hc, k = 4)
prediction
```

```
## 10067211 10067219 10067220 10067221 10067222 10067223 10067224 10067225
##        1        2        1        1        1        1        1        1
## 10067226 10067227 10067228 10067229 10067231 10067233 10067235 10067236
##        1        2        2        2        1        3        3        2
## 10067238 10067241 10067243 10067245 10067246 10067247 10067248 10067251
##        2        2        2        4        1        3        3        3
## 10067261 10067268 10067270 10067273 10067275 10067276 10067488 10067489
##        2        1        2        2        4        4        2        4
## 10067490 10067492 10067493 10067495 10067496 10067497 10067498 10067501
##        2        1        4        2        4        3        2        1
## 10067502 10067515 10067516 10067517 10067826 10067827 10067829 10067830
##        4        2        4        1        2        3        1        2
## 10067836 10067839
##        2        1
```

```
levels <- groups02 %>%
    mutate(level = ifelse(group == "Anus",
        1, "")) %>%
    mutate(level = ifelse(group == "Cervix",
        2, level)) %>%
    mutate(level = ifelse(group == "Head and Neck",
        3, level)) %>%
```

```
    mutate(level = ifelse(group == "Lung",
        4, level)) %>%
    mutate(level = ifelse(group == "Penis",
        5, level)) %>%
    mutate(level = ifelse(group == "Vulva",
        6, level))
target <- levels$level

Reference = groups02$group
Prediction <- cutree(hc, k = 4)
table(Prediction, Reference)
```

```
##           Reference
## Prediction Anus Cervix Head and Neck Lung Penis Vulva
##          1    6      4             1    1     0     4
##          2    6      6             1    0     2     4
##          3    2      0             2    0     3     0
##          4    0      5             2    0     1     0
```

```
confusionMatrix(as.factor(Prediction), as.factor(target),
    mode = "everything")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction 1 2 3 4 5 6
##          1 6 4 1 1 0 4
##          2 6 6 1 0 2 4
##          3 2 0 2 0 3 0
##          4 0 5 2 0 1 0
##          5 0 0 0 0 0 0
##          6 0 0 0 0 0 0
##
## Overall Statistics
##
##                Accuracy : 0.28
##                  95% CI : (0.1623, 0.4249)
##     No Information Rate : 0.3
##     P-Value [Acc > NIR] : 0.6721
##
##                   Kappa : 0.0726
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5 Class: 6
## Sensitivity            0.4286   0.4000   0.3333   0.0000     0.00     0.00
## Specificity            0.7222   0.6286   0.8864   0.8367     1.00     1.00
## Pos Pred Value         0.3750   0.3158   0.2857   0.0000      NaN      NaN
## Neg Pred Value         0.7647   0.7097   0.9070   0.9762     0.88     0.84
## Precision              0.3750   0.3158   0.2857   0.0000       NA       NA
```

```
## Recall                 0.4286   0.4000   0.3333   0.0000     0.00     0.00
## F1                     0.4000   0.3529   0.3077      NaN       NA       NA
## Prevalence             0.2800   0.3000   0.1200   0.0200     0.12     0.16
## Detection Rate         0.1200   0.1200   0.0400   0.0000     0.00     0.00
## Detection Prevalence   0.3200   0.3800   0.1400   0.1600     0.00     0.00
## Balanced Accuracy      0.5754   0.5143   0.6098   0.4184     0.50     0.50
```