RNAseq Analysis E. coli strain MG1655 RNAseq on MG1655 reference

Hernan Lorenzi

12/06/2023

```
pacman::p_load(AnnotationDbi,pheatmap,EnhancedVolcano,ggpubr,DESeq2,stringr,biomaRt,tidyverse,pcaExplore
```

```
#
# --- Function to remove all-zero rows (by adjusting min_total_count the function can filter out rows b
#
remove_all_zero_rows <- function(df, min_total_count = 0){
  df <- df[rowSums(df) > min_total_count,]
  return(df)
}


#
# --- function for PCA plots ---
#
plot_PCA = function(object, color_by="condition",
                    shape_by = 19, ntop=500, size = 3,
                    returnData=FALSE, pcs = c(1,2))
{
  # Check variables are present in object
  intgroup = c(color_by)
  if (shape_by != 19){intgroup <- c(intgroup, shape_by)}
  if (!all(intgroup %in% names(colData(object)))) {
    stop("the argument 'intgroup' should specify columns of colData(dds)")
  }

  # calculate the variance for each gene
  rv <- rowVars(assay(object))

  # select the ntop genes by variance
  select <- order(rv, decreasing=TRUE)[seq_len(min(ntop, length(rv)))]

  # perform a PCA on the data in assay(x) for the selected genes
  pca <- prcomp(t(assay(object)[select,]))

  # the contribution to the total variance for each component
  percentVar <- pca$sdev^2 / sum( pca$sdev^2 )


  intgroup.df <- as.data.frame(colData(object)[, intgroup, drop=FALSE])
```

```
  # add the intgroup factors together to create a new grouping factor
  group <- if (length(intgroup) > 1) {
    factor(apply( intgroup.df, 1, paste, collapse=":"))
  } else {
    colData(object)[[intgroup]]
  }

  # assembly the data for the plot
  d <- data.frame(PC1=pca$x[,pcs[1]], PC2=pca$x[,pcs[2]], group=group, intgroup.df, name=colnames(objec
  colnames(d)[1] <- paste0("PC",pcs[1])
  colnames(d)[2] <- paste0("PC",pcs[2])

  if (returnData) {
    attr(d, "percentVar") <- percentVar[1:2]
    return(d)
  }

  ggplot(data=d, aes_string(x=colnames(d)[1], y=colnames(d)[2], color=color_by, shape=shape_by)) +
    geom_point(size=size) +
    scale_color_lancet() +
    xlab(paste0("PC",pcs[1],": ",round(percentVar[pcs[1]] * 100),"% variance")) + # fixed
    ylab(paste0("PC",pcs[2],": ",round(percentVar[pcs[2]] * 100),"% variance")) + # fixed
    coord_fixed(ratio = (max(d[,1])-min(d[,1]))/(max(d[,2])-min(d[,2])))
}
```

Load libraries

```
all.star <- read.delim2("../results_MG1655/data/read_counts_MG1655.txt", sep = "\t", header = TRUE, row

format_star <- function(star_file){
  names(star_file) <- names(star_file) %>%
    str_remove_all(pattern = "results.03map_reads.|.Aligned.sortedByCoord.out.bam")
  return(star_file[6:ncol(star_file)])
}

# Format star counts file
all <- format_star(star_file = all.star)

# Make sure read counts are numeric and rounded to 0 decimals
all.tmp <- as.data.frame(lapply(all, function(x){ round(as.numeric(x), digits = 0)} ))
rownames(all.tmp) <- rownames(all)
all <- all.tmp

#Remove all zero rows
all <- remove_all_zero_rows(all, min_total_count = 0)
```

Load read counts data

```
# Load metadata
metadata <- read.delim2("./data/metadata.csv",
                        sep = ",",
```

```
                        header = TRUE,
                        row.names = 1,
                        comment.char = c("#") )

# sort all columns based on metadata row names
all <- all %>% dplyr::select(rownames(metadata))

# Add total read counts and sample id columns to metadata
metadata$read_counts <- colSums(all)

# Add "Sample_name" as column in metadata
metadata$sample_name <- rownames(metadata)

# edit treatment column
metadata$treatment <- str_remove(metadata$treatment, pattern = "Grown at ")

# Kepp columns of interest
metadata <- metadata %>% dplyr::select(c("genotype","treatment","read_counts","sample_name"))

# change label for mutant
metadata[metadata$genotype == "RpoD D445V mutant","genotype"] <- "mutant"

# Add column combining genotype and treatment
metadata$group <- paste(metadata$treatment,metadata$genotype, sep = "_")
```

Make metadata table from 'all'

```
# Function to normalize by TPMs based on transcript length
# Normalize counts to TPMs
# Fetch exon length from STAR read counts file
normalize_by_TPM <- function(counts.df, gene_length) {

  # Calculate transcript length in Kb
  transcript_lengths <- gene_length / 1000
  transcript_lengths <- subset(transcript_lengths, rownames(transcript_lengths) %in% rownames(counts.df)

  # Eliminate gene IDs from counts.df without transcript length info in transcript_lengths
  #transcript_lengths <- transcript_lengths[transcript_lengths$Category %in% rownames(counts.df),]
  #counts.df <- counts.df[transcript_lengths$Category,]

  # Sort transcripts_length df by rownames of counts.df
  transcript_lengths <- transcript_lengths[rownames(counts.df),, drop=F]

  # See reference for formula
  # https://btep.ccr.cancer.gov/question/faq/what-is-the-difference-between-rpkm-fpkm-and-tpm/
  x.df <- apply(counts.df,
                MARGIN = 2,
                FUN = function(x){
                                  reads_per_kb <- x/transcript_lengths$Length
                                  pmsf <- sum(reads_per_kb) / 1e6
                                  reads_per_kb/pmsf
                                  }
                )
```

```
  return(x.df)
}

# Using annotation version GRCm39 (current)
all.tpm <- normalize_by_TPM(counts.df = all,
                            gene_length = dplyr::select(all.star, c("Length")))
```

Normalize data to TPMs to run some comparative analysis across samples

Analysis of expression data using DESeq2

```
# Convert metadata to factors
for (variable in c("genotype", "treatment","sample_name","group")){
  metadata[,variable] <- as.factor(str_replace_all(metadata[,variable], pattern = " ", replacement = "_
}
```

Analysis of Dataset

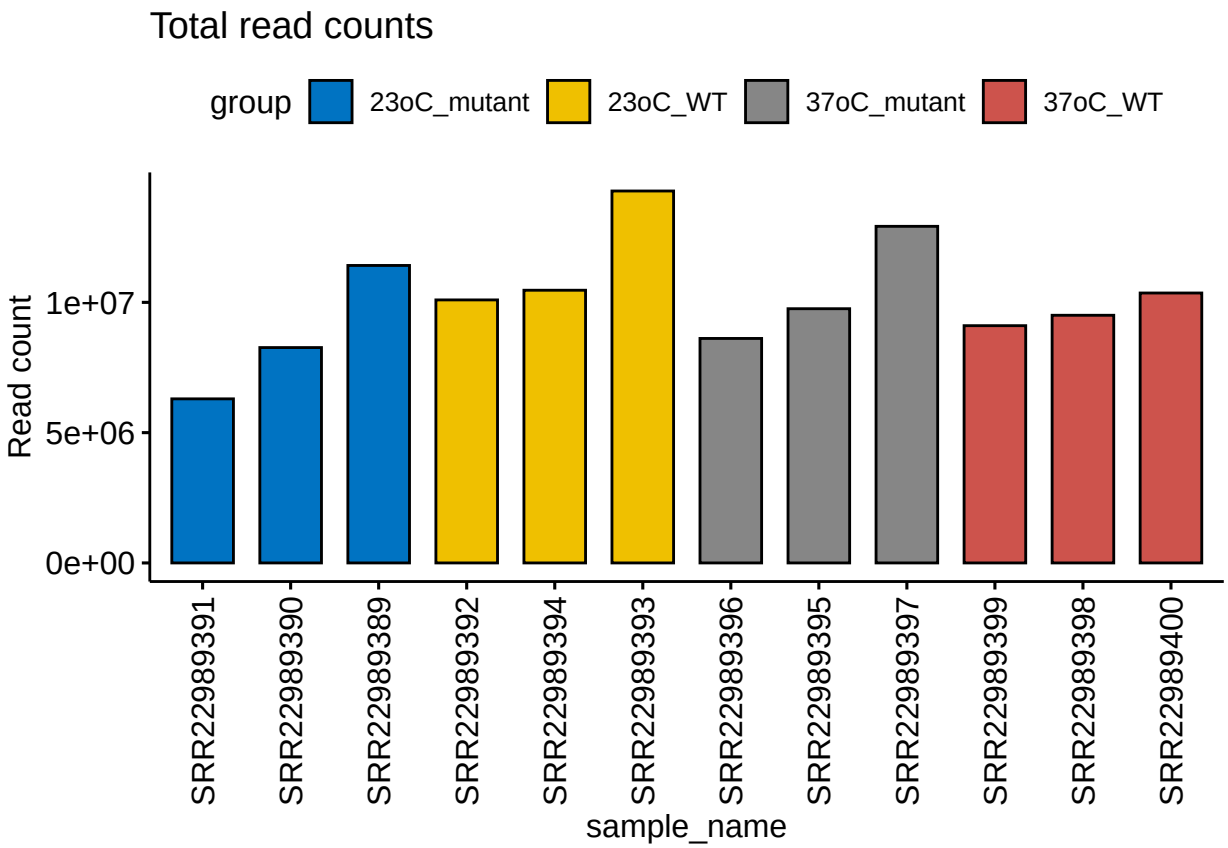
```
# Generate DESeq2 object for NS and ST condition ONLY. We could potentially add Read_counts as either a

dir.create(path = "./Plots", showWarnings = FALSE)

# Create DESeq object
dds.all <- DESeqDataSetFromMatrix(countData = all,
                                  colData = metadata,
                                  design = ~ group)
```

```
## converting counts to integer mode
```

```
# Plot total reads per sample using barchar
p <- ggbarplot(data = metadata,
          x = "sample_name",
          y = "read_counts",
          x.text.angle = 90,
          fill = "group",
          title = "Total read counts",
          ylab = "Read count",
          sort.by.groups = TRUE,
          palette = "jco",
          sort.val = "asc")
ggsave2("Plots/barplot_read_counts.pdf", plot = p)
```

```
## Saving 6.5 x 4.5 in image
```

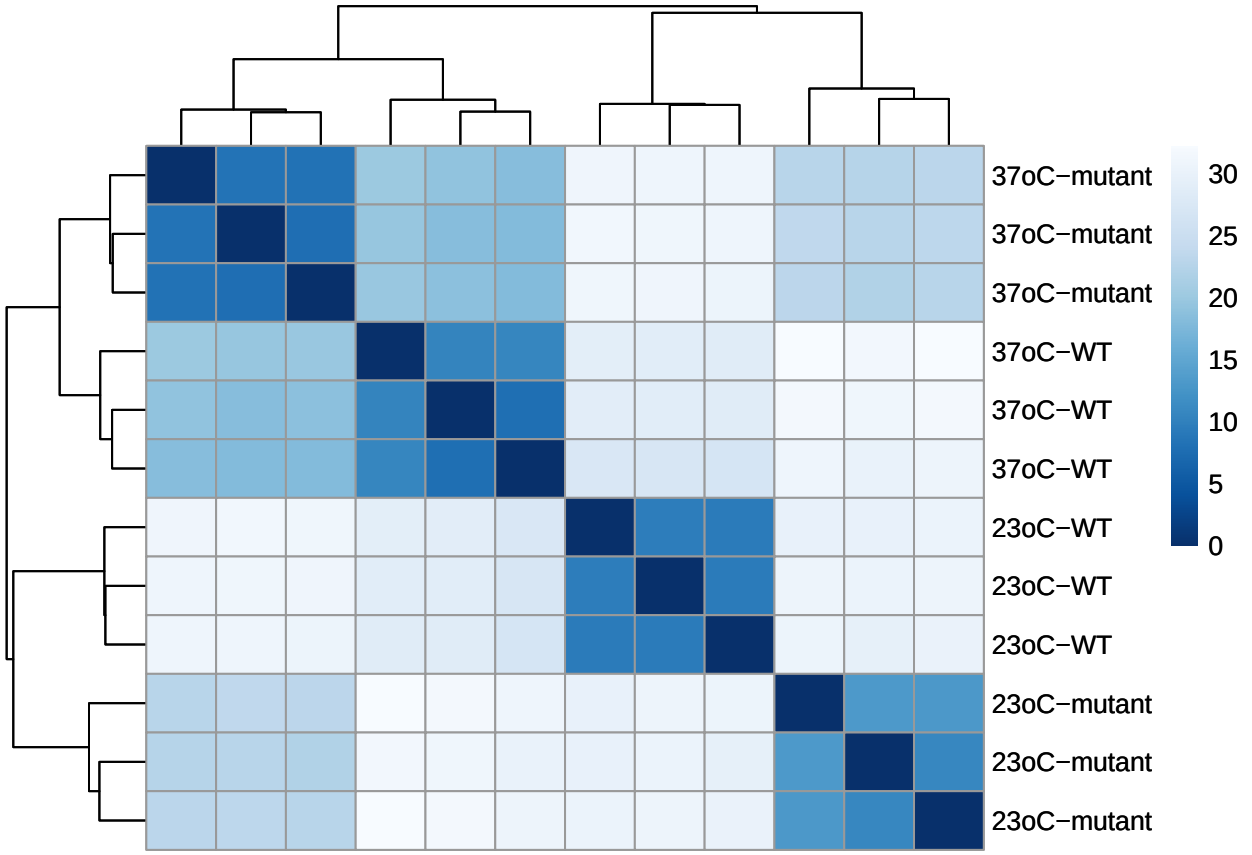
```
print(p)
```




```
# Normalize counts
vsd.one <- vst(dds.all, blind=FALSE)
rlog.one <- rlog(dds.all, blind=FALSE)

# Keep genes with at least 20 reads total across samples
keep <- rowSums(counts(dds.all)) >= 20
dds.all <- dds.all[keep,]

# Calculate distances between samples
sampleDists <- dist(t(assay(vsd.one)))

# Plot inter-sample distances
old.par <- par(no.readonly=T)

sampleDistMatrix <- as.matrix(sampleDists)
rownames(sampleDistMatrix) <- paste(rlog.one$treatment, rlog.one$genotype, sep="-")
colnames(sampleDistMatrix) <- NULL
colors <- colorRampPalette( rev(brewer.pal(9, "Blues")) )(255)
p.pheatmap <- pheatmap(sampleDistMatrix,
         clustering_distance_rows=sampleDists,
         clustering_distance_cols=sampleDists,
         col=colors)

ggsave2(filename = "unsupervised_clustering_rnaseq_profile_20plus_reads.pdf", plot = p.pheatmap, path =
```

```
## Saving 6.5 x 4.5 in image
```

```
print(p.pheatmap)
```




```
dds_res <- list()

dds_res <- dds.all #[ , dds.all$Tissue=="all_data"]

rlog_res <- list()
rlog_res <- rlog(dds_res, blind=FALSE)

# PCA
rlog.one <- rlog_res

# PC1 - PC2
principal_components <- c(1,2)
pca_12.p <- plot_PCA(object = rlog.one,color_by = "genotype", shape_by = "treatment", returnData = FALSE
```

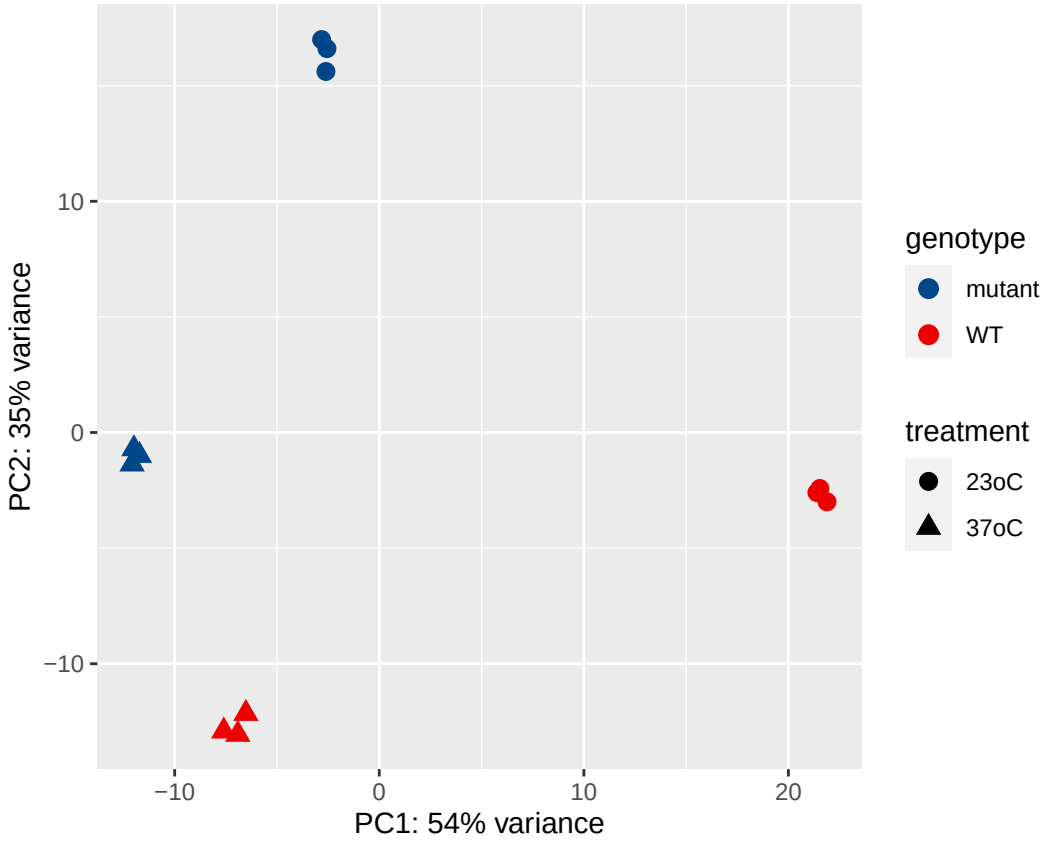
```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation idioms with `aes()`.
## i See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
# PC2 - PC3
principal_components <- c(2,3)
pca_23.p <- plot_PCA(object = rlog.one,color_by = "genotype", shape_by = "treatment", returnData = FALSE
```

```
# PC1 - PC3
principal_components <- c(1,3)
pca_13.p <- plot_PCA(object = rlog.one,color_by = "genotype", shape_by = "treatment", returnData = FALSE
ggsave(paste0("Plots/pca_PC12_Group.pdf"), plot = pca_12.p)
```

```
## Saving 6.5 x 4.5 in image
```

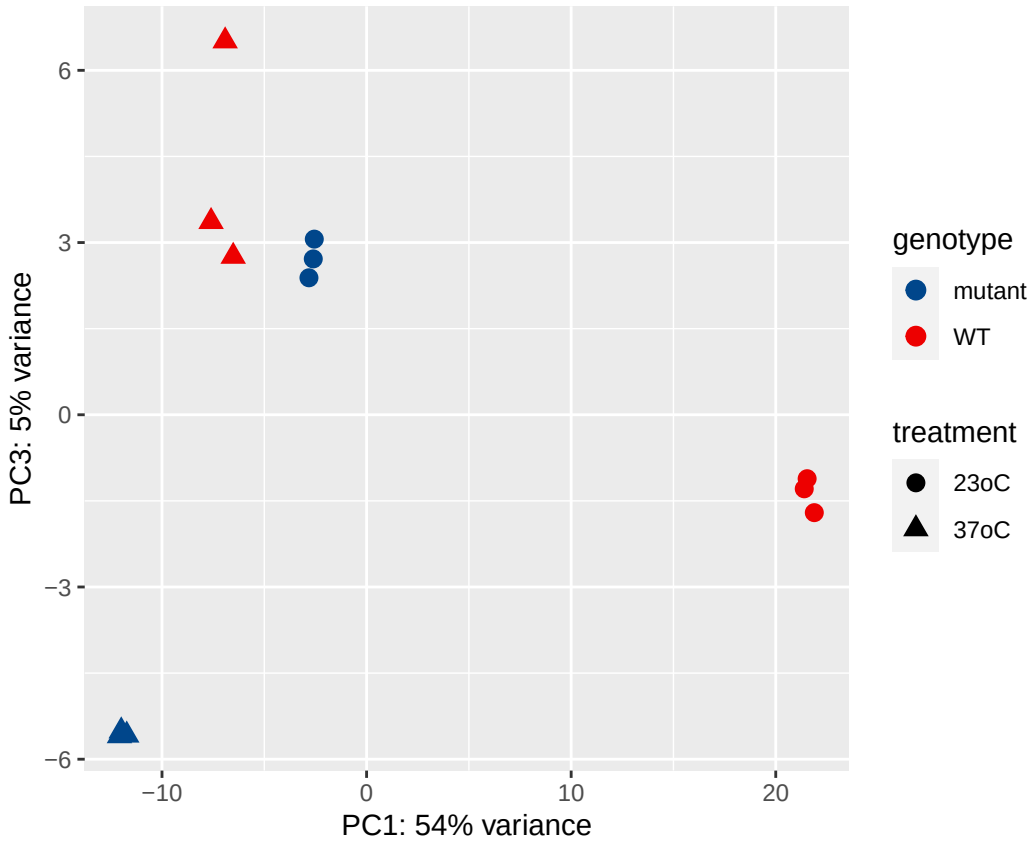
```
print(pca_12.p)
```




```
ggsave(paste0("Plots/pca_PC13_Group.pdf"), plot = pca_13.p)
```

```
## Saving 6.5 x 4.5 in image
```

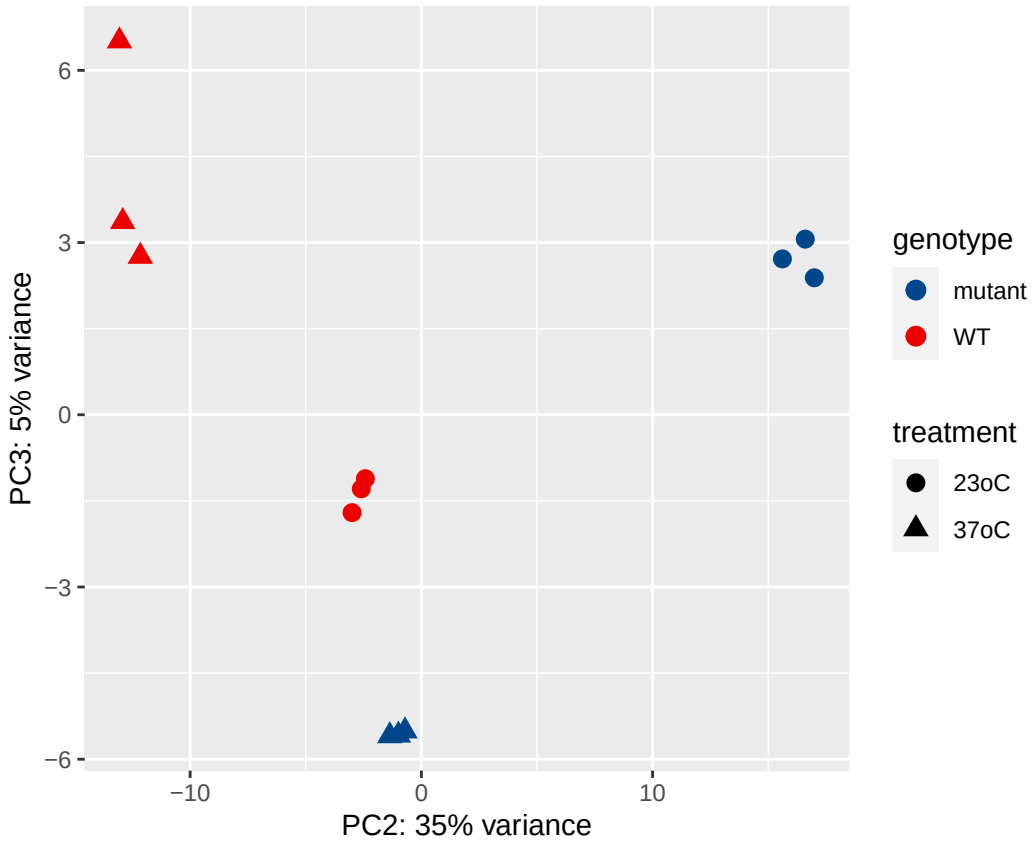
```
print(pca_13.p)
```




```
ggsave(paste0("Plots/pca_PC23_Group.pdf"), plot = pca_23.p)
```

```
## Saving 6.5 x 4.5 in image
```

```
print(pca_23.p)
```



PCA analysis shows that samples separate by genotype and treatment.

resultsNames(dds)

```
# Keep genes with at least 10 reads total across samples
keep <- rowSums(counts(dds_res)) >= 20
dds_res <- dds_res[keep,]
```

Filtering out poorly-expressed genes (less than 20 reads across all samples)

```
ensembl_to_symbol <- read.delim(file = "./data/gene_names.txt", col.names = c("Ensembl_ID","gene_name")

# Save sorted files as a list
DE_results <- list()
geneids.DE <- list()

# Define function for processing and saving result tables
sort_and_write_res_table <- function(result_table, file_name){
  dir.create(path = "./DE", showWarnings = FALSE)
  # Sort genes by (padj)
  result_table_sorted <- result_table[order(result_table$padj, decreasing = FALSE),]
  # Add gene symbols
  gene_list <- rownames(result_table_sorted)
  symbol_list <- ensembl_to_symbol$gene_name[match(gene_list, ensembl_to_symbol$Ensembl_ID)]
  df <-as.data.frame(cbind(result_table_sorted, Gene_name = symbol_list))
```

```
  # Write sorted table to file
  write.table(df, file = paste0("./DE/",file_name,".txt"),
              sep = "\t", col.names=NA)
  return(df)
}
```

```
# Calculate DE for all_data samples
dds_res$group <- relevel(dds_res$group, "37oC_WT")
dds_res <- DESeq(dds_res)
```

Using groups instead of interactions

```
## estimating size factors
```

```
## estimating dispersions
```

```
## gene-wise dispersion estimates
```

```
## mean-dispersion relationship
```

```
## final dispersion estimates
```

```
## fitting model and testing
```

```
my_contrasts <- resultsNames(dds_res)


# Using lfcShrink instead of results to reduce high Log2FC bias of genes with low expression
# 37oC mutant vs WT
res_mut_vs_WT_37C <- lfcShrink(dds_res, coef = my_contrasts[4], type = "ashr", )
```

```
## using 'ashr' for LFC shrinkage. If used in published research, please cite:
##     Stephens, M. (2016) False discovery rates: a new deal. Biostatistics, 18:2.
##     https://doi.org/10.1093/biostatistics/kxw041
```

```
res_mut_vs_WT_23C <- lfcShrink(dds_res, contrast = c("group", "23oC_mutant", "23oC_WT"), type = "ashr",
```

```
## using 'ashr' for LFC shrinkage. If used in published research, please cite:
##     Stephens, M. (2016) False discovery rates: a new deal. Biostatistics, 18:2.
##     https://doi.org/10.1093/biostatistics/kxw041
```

```
summary(res_mut_vs_WT_37C, alpha = 0.05)
```

```
##
## out of 4525 with nonzero total read count
## adjusted p-value < 0.05
## LFC > 0 (up)       : 630, 14%
## LFC < 0 (down)     : 660, 15%
## outliers [1]       : 7, 0.15%
## low counts [2]     : 790, 17%
## (mean count < 9)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

```
summary(res_mut_vs_WT_23C, alpha = 0.05)
```

```
##
## out of 4525 with nonzero total read count
## adjusted p-value < 0.05
```

```
## LFC > 0 (up)       : 829, 18%
## LFC < 0 (down)     : 852, 19%
## outliers [1]       : 7, 0.15%
## low counts [2]     : 614, 14%
## (mean count < 7)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

```
# Sort results by Log2FC
res_mut_vs_WT_37C_sorted <- sort_and_write_res_table(result_table = res_mut_vs_WT_37C, file_name = paste
res_mut_vs_WT_23C_sorted <- sort_and_write_res_table(result_table = res_mut_vs_WT_23C, file_name = paste
```

```
table_counts_normalized <- counts(dds_res, normalized=TRUE)
write.table(x = as.data.frame(table_counts_normalized), file = "read_counts_deseq2_normalized.txt", sep
```

```
genes_of_interest <- c("metE","ampC","fucI","aceB","shiA", "ybgD", "mlaA", "cysZ", "acrZ", "bcr","sppA"
```

Genes of interest

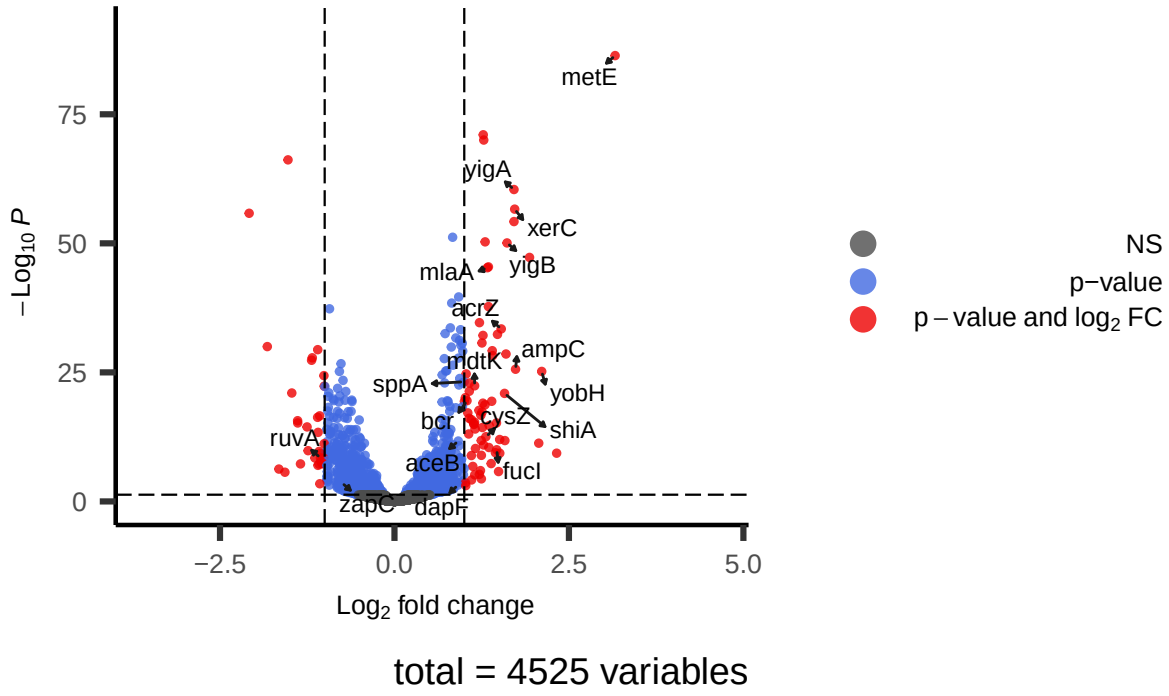
```
volcano_plot_with_ids <- function(res.tmp, log_scale = FALSE, gene_list){
  vp <- EnhancedVolcano(res.tmp,
                        lab = res.tmp$Gene_name,
                        x = 'log2FoldChange',
                        y = 'padj',
                        pCutoff = 0.05,
                        FCcutoff = 1,
                        pointSize = 1,
                        colAlpha = 4/5,
                        labSize = 3,  # Controls labels size
                        labCol = "black",
                        title = '',
                        titleLabSize = 10,
                        subtitle = '', # add subtitle here
                        subtitleLabSize = 10,
                        legendPosition = 'right',
                        legendLabSize = 10,
                        legendIconSize = 4.0,
                        axisLabSize = 10,
                        drawConnectors = TRUE,
                        selectLab = gene_list, # vector of gene symbols to label on volcanoplot
                        boxedLabels = FALSE,
                        gridlines.major = FALSE,
                        gridlines.minor = FALSE,
                        hlineCol = "gray", vlineCol = "gray"
  )

  #theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank())

  if (log_scale){
    vp <- vp + scale_x_log10()
  }
```

```
  return(vp)
}


vp1 <- volcano_plot_with_ids(res.tmp = res_mut_vs_WT_37C_sorted,
                             log_scale = FALSE,
                             gene_list = genes_of_interest)

ggsave(filename = paste0("./Plots/mut_vs_WT_37C_VolcanoPlot.pdf"),
       plot = vp1, width = 6, height = 6)

vp2 <- volcano_plot_with_ids(res.tmp = res_mut_vs_WT_23C_sorted,
                             log_scale = FALSE,
                             gene_list =genes_of_interest
                             )
```

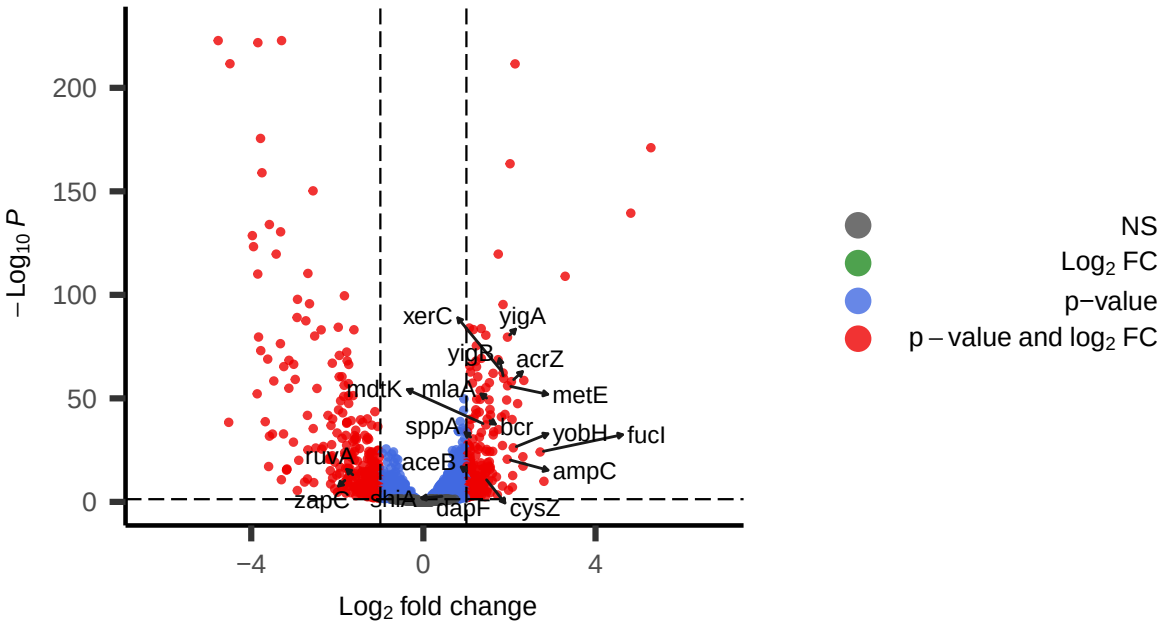
Generate volcano plots

```
## Warning: One or more p-values is 0. Converting to 10^-1 * current lowest
## non-zero p-value...
```

```
ggsave(filename = paste0("./Plots/mut_vs_WT_23C_VolcanoPlot.pdf"),
       plot = vp2, width = 6, height = 6)

print(vp1)
```

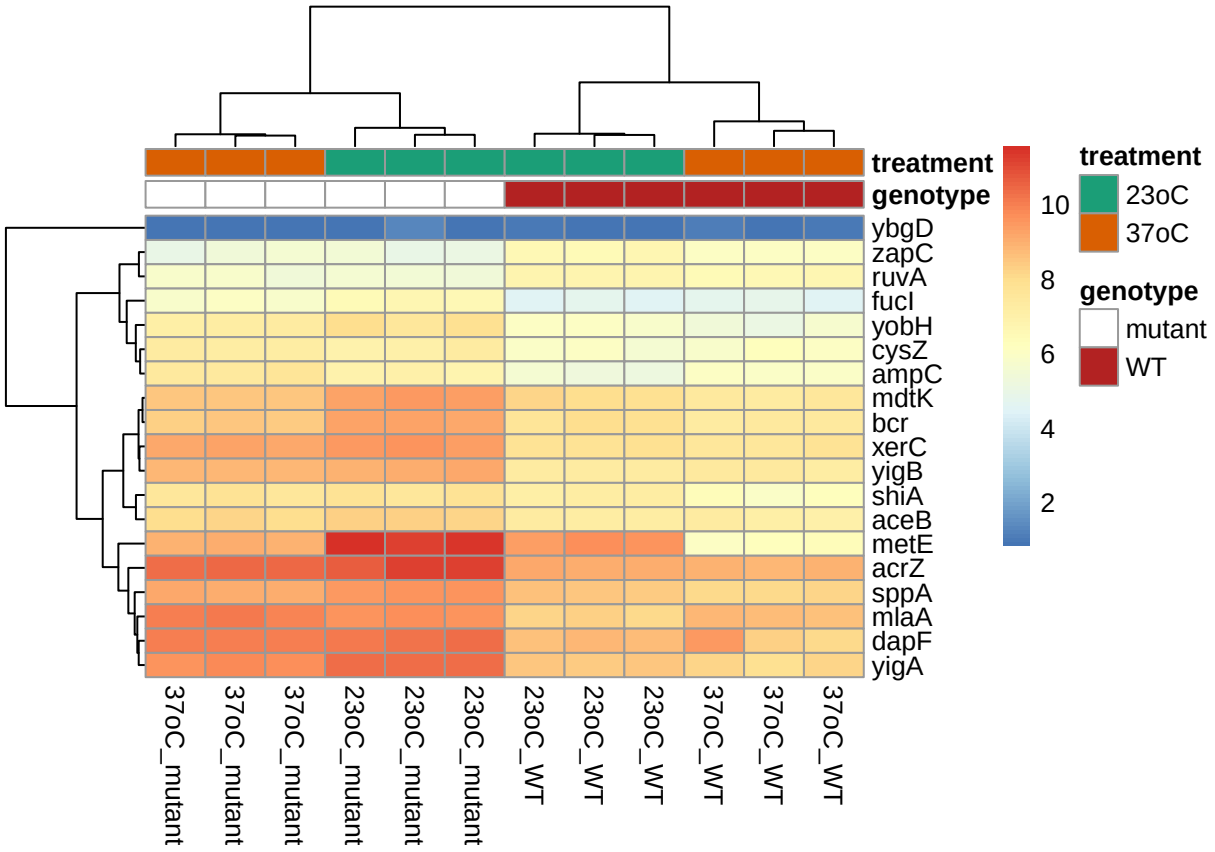



```
print(vp2)
```



total = 4525 variables

```
#genes_of_interest.ensmbl <- rownames(head(assay(rlog.one)))
genes_of_interest.names<- subset(ensembl_to_symbol, gene_name %in% genes_of_interest)

# Specify colors
ann_colors = list(
    genotype = c(mutant = "white", WT = "firebrick"),
    treatment = c("23oC" = "#1B9E77", "37oC" = "#D95F02"))

annot_col <- as.data.frame(dplyr::select(metadata,c("genotype","treatment")))

p1.heatmap <- pheatmap(assay(rlog.one)[genes_of_interest.names$Ensembl_ID,],
          cluster_rows=T,
          show_rownames=TRUE,
          cluster_cols=T,
          annotation_col = annot_col,
         labels_row = genes_of_interest.names$gene_name,
         labels_col = metadata$group,
         annotation_colors = ann_colors)
```



Plot heatmaps

```
ggsave2(filename = "mut_vs_wt_heatmap.pdf", plot = p1.heatmap, path = "./Plots", width = 7, height = 5)
```

Get DESeq-normalized counts

```
table_counts_normalized <- counts(dds_res, normalized=TRUE)
write.table(x = as.data.frame(table_counts_normalized), file = "read_counts_deseq2_normalized.txt", sep
```

```
print(sessionInfo())
```

```
## R version 4.3.1 (2023-06-16)
## Platform: x86_64-apple-darwin20 (64-bit)
## Running under: macOS Ventura 13.6.3
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/lib/libRlapack.dylib;  LAPACK
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## time zone: America/New_York
## tzcode source: internal
##
## attached base packages:
## [1] grid      stats4    stats     graphics  grDevices utils     datasets
## [8] methods   base
```

```
##
## other attached packages:
##  [1] ggraph_2.1.0                broom_1.0.5
##  [3] ggupset_0.3.0               enrichplot_1.22.0
##  [5] cowplot_1.1.1               msigdbr_7.5.1
##  [7] RColorBrewer_1.1-3          viridis_0.6.4
##  [9] viridisLite_0.4.2           ggsci_3.0.0
## [11] GOSemSim_2.28.0             clusterProfiler_4.10.0
## [13] VennDiagram_1.7.3           futile.logger_1.4.3
## [15] pcaExplorer_2.28.0          lubridate_1.9.3
## [17] forcats_1.0.0               dplyr_1.1.4
## [19] purrr_1.0.2                 readr_2.1.4
## [21] tidyr_1.3.0                 tibble_3.2.1
## [23] tidyverse_2.0.0             biomaRt_2.58.0
## [25] stringr_1.5.1               DESeq2_1.42.0
## [27] SummarizedExperiment_1.32.0 MatrixGenerics_1.14.0
## [29] matrixStats_1.1.0           GenomicRanges_1.54.1
## [31] GenomeInfoDb_1.38.1         ggpubr_0.6.0
## [33] EnhancedVolcano_1.20.0      ggrepel_0.9.4
## [35] ggplot2_3.4.4               pheatmap_1.0.12
## [37] AnnotationDbi_1.64.1        IRanges_2.36.0
## [39] S4Vectors_0.40.2            Biobase_2.62.0
## [41] BiocGenerics_0.48.1
##
## loaded via a namespace (and not attached):
##   [1] fs_1.6.3                bitops_1.0-7            HDO.db_0.99.1
##   [4] httr_1.4.7              webshot_0.5.5           doParallel_1.0.17
##   [7] Rgraphviz_2.46.0        tools_4.3.1             backports_1.4.1
##  [10] utf8_1.2.4              R6_2.5.1                DT_0.30
##  [13] lazyeval_0.2.2          withr_2.5.2             prettyunits_1.2.0
##  [16] gridExtra_2.3           textshaping_0.3.7       cli_3.6.1
##  [19] pacman_0.5.1            formatR_1.14            TSP_1.2-4
##  [22] scatterpie_0.2.1        labeling_0.4.3          topGO_2.54.0
##  [25] SQUAREM_2021.1          genefilter_1.84.0       mixsqp_0.3-48
##  [28] systemfonts_1.0.5       yulab.utils_0.1.0       gson_0.1.0
##  [31] DOSE_3.28.1             AnnotationForge_1.44.0  invgamma_1.1
##  [34] limma_3.58.1            rstudioapi_0.15.0       RSQLite_2.3.3
##  [37] gridGraphics_0.5-1      generics_0.1.3          GOstats_2.68.0
##  [40] crosstalk_1.2.1         car_3.1-2               dendextend_1.17.1
##  [43] GO.db_3.18.0            Matrix_1.6-4            fansi_1.0.5
##  [46] abind_1.4-5             lifecycle_1.0.4         yaml_2.3.7
##  [49] carData_3.0-5           qvalue_2.34.0           SparseArray_1.2.2
##  [52] BiocFileCache_2.10.1    blob_1.2.4              promises_1.2.1
##  [55] crayon_1.5.2            shinydashboard_0.7.2    lattice_0.21-8
##  [58] annotate_1.80.0         KEGGREST_1.42.0         pillar_1.9.0
##  [61] knitr_1.45              fgsea_1.28.0            codetools_0.2-19
##  [64] fastmatch_1.1-4         glue_1.6.2              ggfun_0.1.3
##  [67] data.table_1.14.8       treeio_1.27.0.001       vctrs_0.6.5
##  [70] png_0.1-8               gtable_0.3.4            assertthat_0.2.1
##  [73] cachem_1.0.8            xfun_0.41               S4Arrays_1.2.0
##  [76] mime_0.12               tidygraph_1.2.3         survival_3.5-5
##  [79] seriation_1.5.3         iterators_1.0.14        statmod_1.5.0
##  [82] ellipsis_0.3.2          nlme_3.1-162            Category_2.68.0
##  [85] ggtree_3.10.0           bit64_4.0.5             threejs_0.3.3
```

```
##  [88] progress_1.2.3          filelock_1.0.2          irlba_2.3.5.1
##  [91] colorspace_2.1-0        DBI_1.1.3               tidyselect_1.2.0
##  [94] bit_4.0.5               compiler_4.3.1          curl_5.1.0
##  [97] graph_1.80.0            SparseM_1.81            xml2_1.3.6
## [100] DelayedArray_0.28.0     plotly_4.10.3           shadowtext_0.1.2
## [103] scales_1.3.0            RBGL_1.78.0             NMF_0.26
## [106] rappdirs_0.3.3          digest_0.6.33           shinyBS_0.61.1
## [109] rmarkdown_2.25          ca_0.71.1               XVector_0.42.0
## [112] htmltools_0.5.7         pkgconfig_2.0.3         base64enc_0.1-3
## [115] highr_0.10              dbplyr_2.4.0            fastmap_1.1.1
## [118] rlang_1.1.2             htmlwidgets_1.6.4       shiny_1.8.0
## [121] farver_2.1.1            jsonlite_1.8.8          BiocParallel_1.36.0
## [124] RCurl_1.98-1.13         magrittr_2.0.3          ggplotify_0.1.2
## [127] GenomeInfoDbData_1.2.11 patchwork_1.1.3         munsell_0.5.0
## [130] Rcpp_1.0.11             babelgene_22.9          ape_5.7-1
## [133] stringi_1.8.2           zlibbioc_1.48.0         MASS_7.3-60
## [136] plyr_1.8.9              parallel_4.3.1          Biostrings_2.70.1
## [139] graphlayouts_1.0.2      splines_4.3.1           hms_1.1.3
## [142] locfit_1.5-9.8          igraph_1.5.1            ggsignif_0.6.4
## [145] rngtools_1.5.2          reshape2_1.4.4          futile.options_1.0.1
## [148] XML_3.99-0.16           evaluate_0.23           lambda.r_1.2.4
## [151] BiocManager_1.30.22     tzdb_0.4.0              foreach_1.5.2
## [154] tweenr_2.0.2            httpuv_1.6.13           polyclip_1.10-6
## [157] heatmaply_1.5.0         ashr_2.2-63             gridBase_0.4-7
## [160] ggforce_0.4.1           xtable_1.8-4            tidytree_0.4.5
## [163] rstatix_0.7.2           later_1.3.2             ragg_1.2.6
## [166] truncnorm_1.0-9         aplot_0.2.2             memoise_2.0.1
## [169] registry_0.5-1          cluster_2.1.4           timechange_0.2.0
## [172] shinyAce_0.4.2          GSEABase_1.64.0
```