PowerPuff Genes Project

2025-03-17

Contents

Getting Started

Librarires prerequisites

- IRanges
- dplyr
- tidyverse
- ...

Data files

- salmon counts
- salmon tpm
- maybe more files, maybe not

RNA-seq differential expression analysis

Define variables

```r
COUNTS_PATH      <- "data/NF3-14_rep1/salmon.merged.gene_counts.tsv"
TPM_PATH         <- "data/NF3-14_rep1/salmon.merged.gene_tpm.tsv"
FILTER_TRESHOLD  <- 5
P_VALUE_FILTER   <- 0.05
LOG2_FOLD_FILTER <- 2
```

Counts analysis

```r
# Read raw count data and make dictionary mapping gene id with gene name
counts_orig  <- read.table(COUNTS_PATH, header=TRUE, row.names=1)
g2s          <- data.frame(gene_id = rownames(counts_orig),
                           gene_name = counts_orig[ , 1])
write.csv(g2s, "results/g2s.csv")

# Remove gene name column from counts and convert counts to numerical rounded matrix
counts           <- counts_orig %>% select(-gene_name)
counts_matrix    <- as.matrix(counts)
counts_matrix    <- round(counts_matrix)
```

```r
# Filter genes with no counts for any of the samples
counts_filtered  <- counts_matrix[rowSums(counts_matrix) > FILTER_TRESHOLD, ]

save(counts_orig, counts, counts_filtered, g2s, file = "results/counts_files.RData")
```

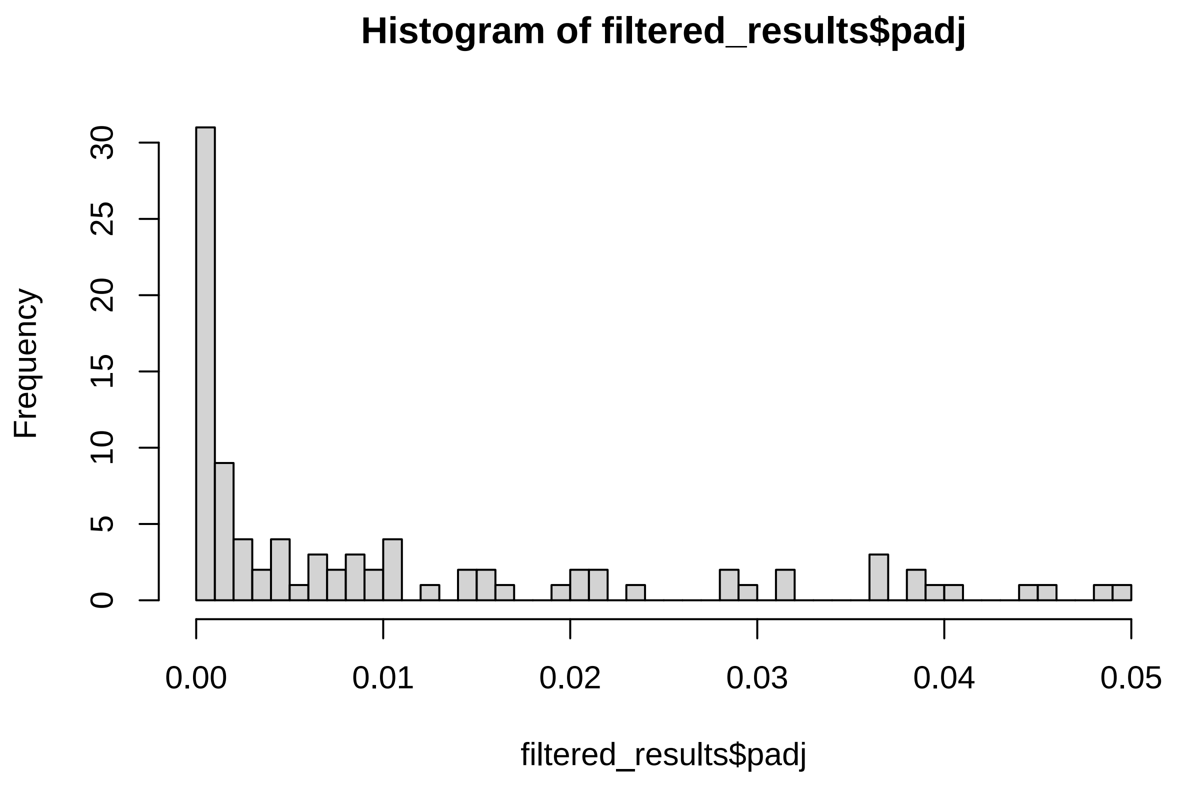
```r
# Make a column from the titles of the columns of the counts_matrix table
deseq_samples  <- data.frame(sample_id = colnames(counts))

# Get the time and replicate values from the sample names in deseq_samples
split_values     <- strsplit(deseq_samples$sample_id, "_")
time_values      <- sapply(split_values, function(x) x[[2]])
replicate_values <- sapply(split_values, function(x) x[[3]])

# Add time and replicate values as columns to deseq_samples and factor them
deseq_samples$time_point <- time_values
deseq_samples$replicate  <- replicate_values
deseq_samples$time_point <- factor(deseq_samples$time_point)
deseq_samples$replicate  <- factor(deseq_samples$replicate)

# Testing whether sample sheet and counts are arranged properly
stopifnot(all(colnames(counts) == rownames(deseq_samples$sample_id)))

save(deseq_samples, file = "results/deseq_samples.RData")
```

```r
# Prepare DESeq dataset
dds <- DESeqDataSetFromMatrix(countData = counts_filtered,
                              colData = deseq_samples,
                              design = ~ time_point)

# Run DESeq and get the names for the comparisons
dds            <- DESeq(dds)
results_names  <- resultsNames(dds)
results_names  <- results_names[results_names != "Intercept"]

# Regularized Log transformation (rlog) stabilizes variance for downstream visualization or clustering.
#rlog_counts <- rlog(dds, blind = TRUE)
#rlog_counts_matrix <- assay(rlog_counts)
#write_rds(rlog_counts_matrix, "results/rlog_counts.rds")

save(dds, file = "results/dds.RData")
```

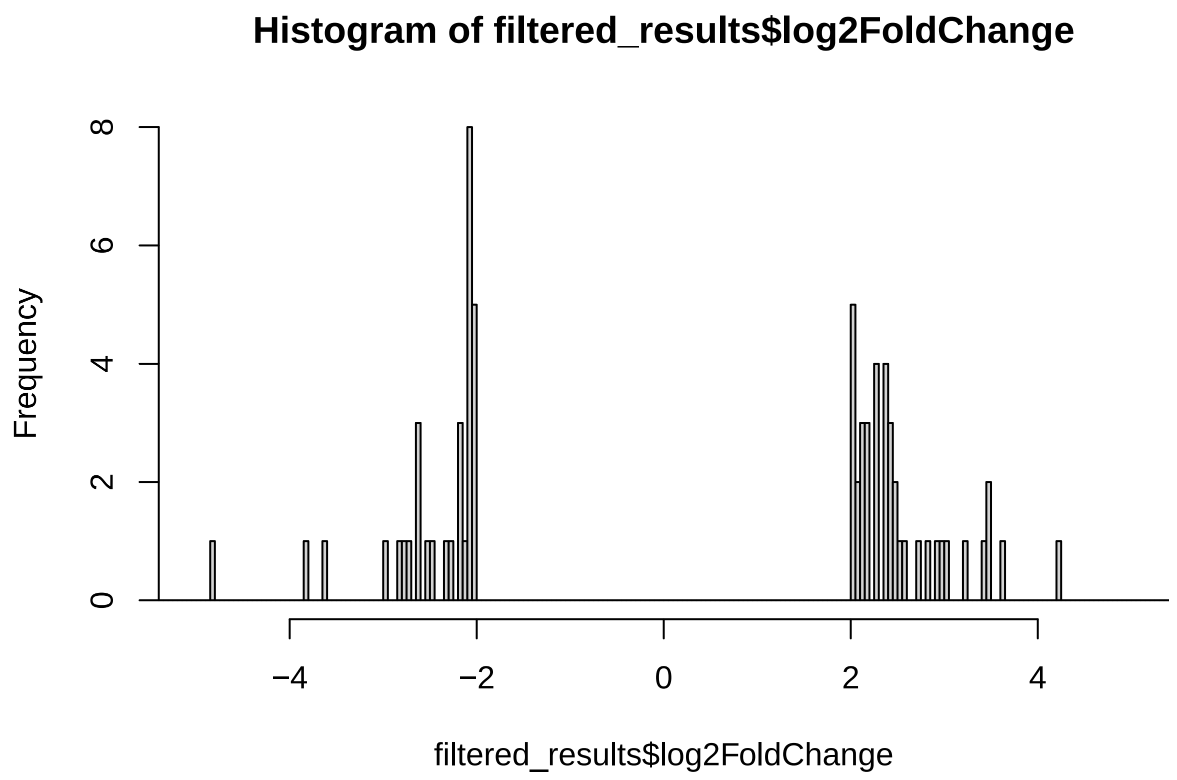
```r
# Create empty df to store results values
results <- data.frame("gene_id" = character(),
                      "baseMean" = numeric(),
                      "log2FoldChange" = numeric(),
                      "lfcSE" = numeric(),
                      "stat" = numeric(),
                      "pvalue" = numeric(),
                      "padj" = numeric(),
                      "gene_name" = character(),
                      "result_name" = character())

# Loop to get the results of each comparison from dds object and make main res_df with all of them
for(i in 1:length(results_names)) {
  results_name <- results_names[i] # get time comparison i
  res <- results(dds, name = results_name) # get DESeq results for time comparison i
  # Temporary df to store the results for each comparison i
  tmp_res_df <- res %>% as.data.frame() %>%
    rownames_to_column("gene_id") %>% merge(g2s) %>% mutate(result_name = results_name)
  # Add the temporary df to the main res_df for each comparison i
  results <- bind_rows(results, tmp_res_df)
}
```

```r
# Filter based on p-value < 0.05 and log2 fold change > 1
filtered_results <- results %>%
  filter(padj < P_VALUE_FILTER, abs(log2FoldChange) > LOG2_FOLD_FILTER)

# Get all gene names that are significant (drop gene name repetitions)
filtered_genes <- as.data.frame(filtered_results$gene_name, collapse = "\n")
filtered_genes <- unique(filtered_genes)
colnames(filtered_genes)[1] <- "gene_name"

# Now let's write this out and do gene enrichment analysis
write.table(filtered_genes["gene_name"], row.names = FALSE, col.names = FALSE, "results/filtered_genes.

save(results, filtered_results, filtered_genes, file = "results/DESeq2_results.RData")

# May wanna take a look at these genes. Start at: https://maayanlab.cloud/Enrichr/
```

```r
# Distribution of baseMean, lfcSE, p-values and fold change
#hist(filtered_results$baseMean, xlim = c(0, 10000), breaks = 500) # May remove this plot
#hist(filtered_results$lfcSE, xlim = c(0,5), breaks = 100) # May remove this histogram as well
hist(filtered_results$padj, xlim = c(0,P_VALUE_FILTER ), breaks = 50)
```



```r
hist(filtered_results$log2FoldChange, xlim = c(-5, 5), breaks = 1000)
```



```r
# T-test comparing baseMean expression of all genes vs filtered genes
basemean_all_genes <- median(results$baseMean)
basemean_fil_genes <- median(filtered_results$baseMean)
t.test(results$baseMean, filtered_results$baseMean)
```

```
##
##  Welch Two Sample t-test
##
## data:  results$baseMean and filtered_results$baseMean
## t = 11.011, df = 100.19, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##   775.569 1116.466
## sample estimates:
## mean of x mean of y
## 1092.5958  146.5784
```

```r
# Intersect by gene_name filtered results table and raw counts
counts_filtered_genes <- counts_orig[filtered_results$gene_id,]
# TODO delete repetitive instances of genes
```

TPM Analysis

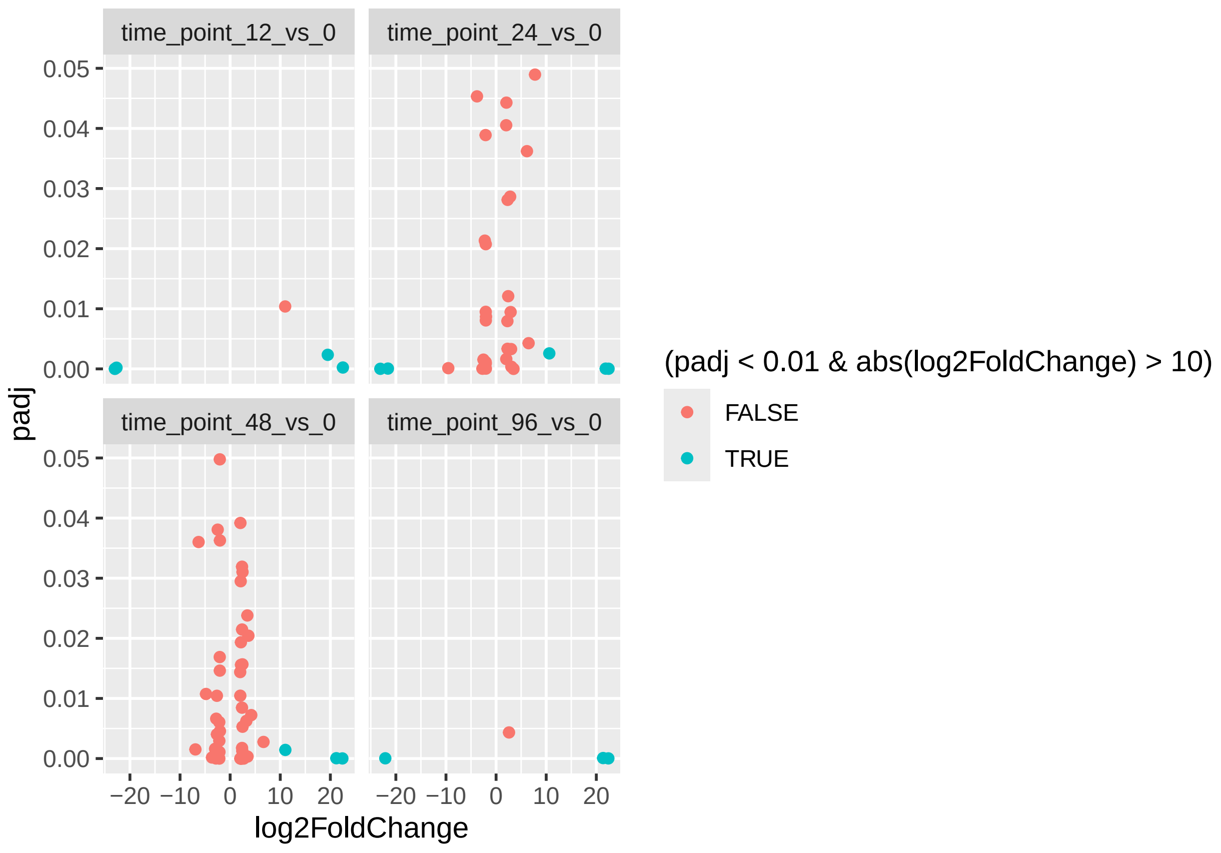
```r
# Read raw tpm data and delete gene name column
tpm_orig  <- read.table(TPM_PATH, header=TRUE, row.names=1)
tpm       <- tpm_orig %>% select(-gene_name)

# Filter genes with no counts for any of the samples
```

```r
tpm_filtered <- tpm[rowSums(tpm) > FILTER_TRESHOLD, ]
```

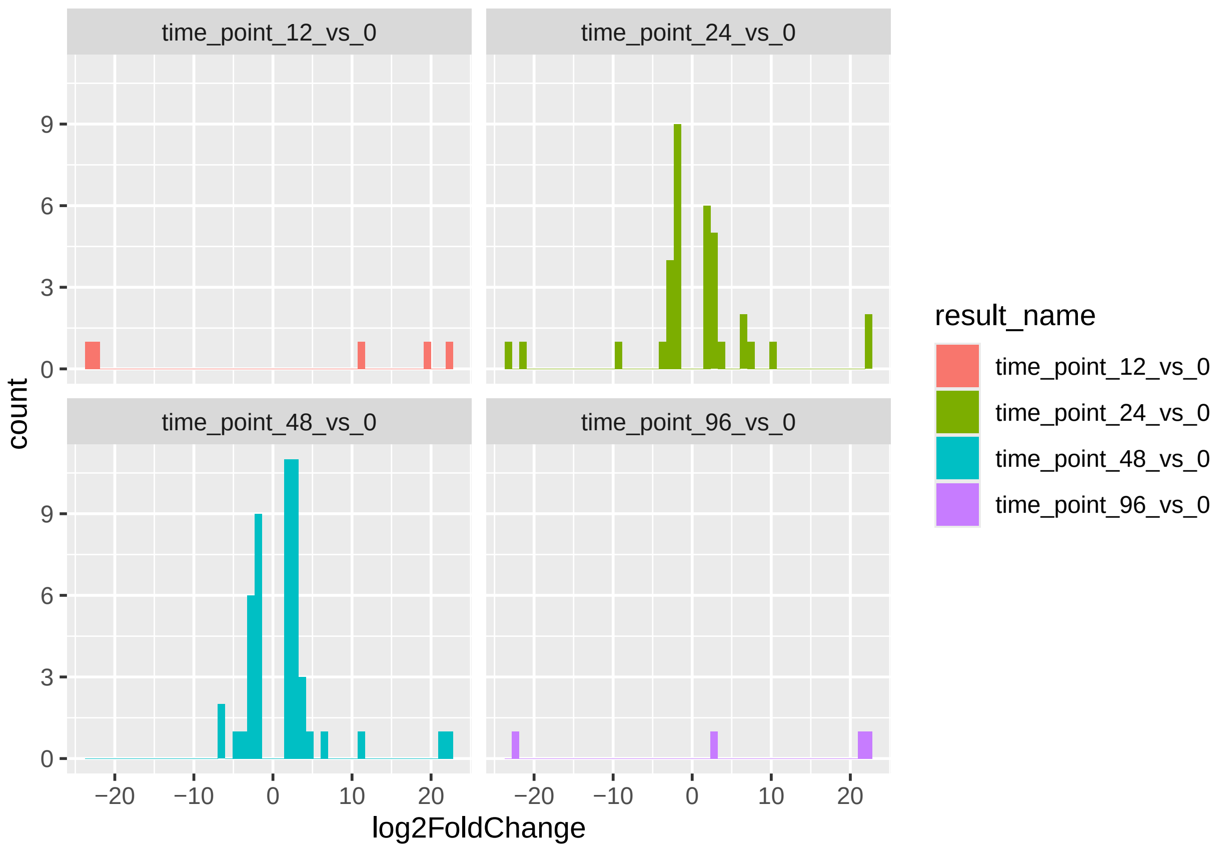
```r
# Loop to calculate avg and sd for replicates at a given time point
time_points <- c("0", "12", "24", "48", "96")
avg_and_sd_values <- list()
for (tp in time_points) {
  cols <- grep(paste0("WT_", tp, "_"), colnames(tpm_filtered))
  avg  <- rowMeans(tpm_filtered[, cols])
  sd   <- apply(tpm_filtered[, cols], 1, sd)
  sd   <- data.frame(sd)
  combined <- cbind(avg, sd)
  avg_and_sd_values <- c(avg_and_sd_values, list(combined))
}

# Convert the list to a data frame and add column names for the respective timepoint
avg_and_sd_values <- do.call(cbind,  avg_and_sd_values)
colnames(avg_and_sd_values) <- paste0(rep(time_points, each = 2), c("_avg", "_sd"))

save(tpm_orig, tpm_filtered, avg_and_sd_values, file = "results/tpm_results.RData" )
```

Image analysis

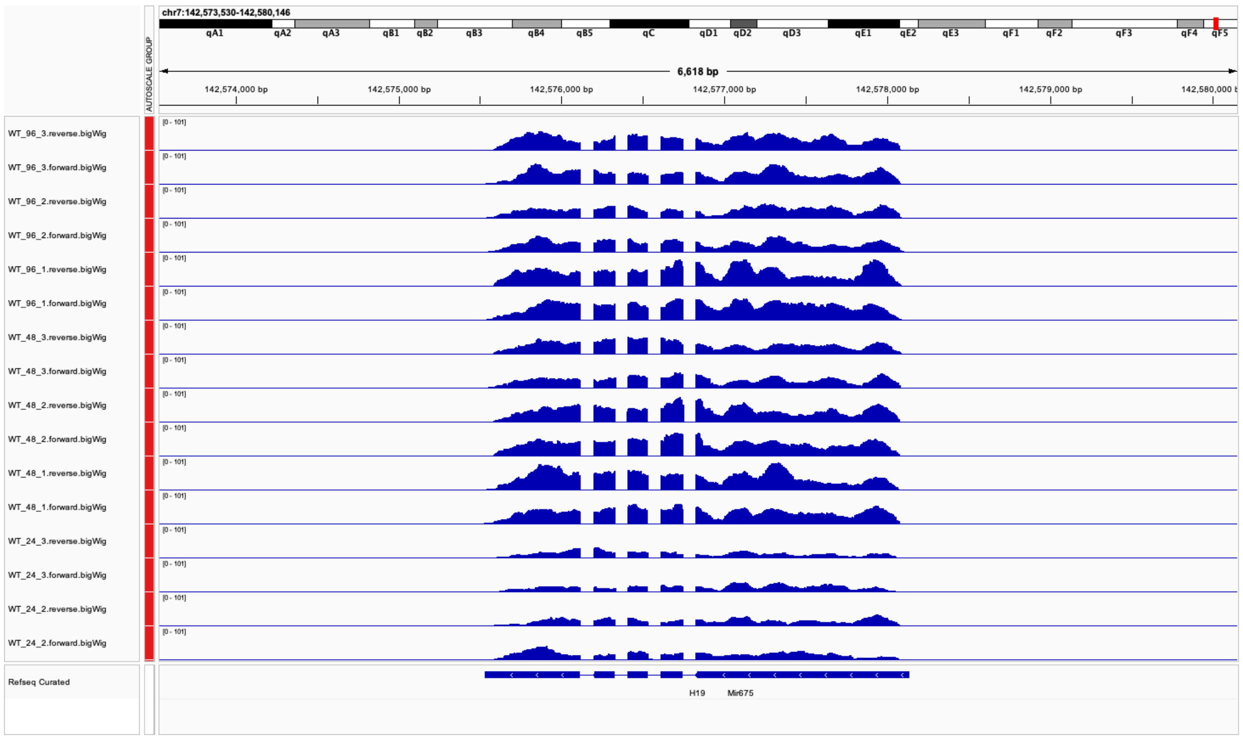
```r
# Delete enviroment and load counts and tpm data
load("results/DESeq2_results.RData")
load("results/tpm_results.RData")

# Volcano plot for log fold change vs padj. TRUE color may be genes worth looking into.
ggplot(filtered_results, aes(x = log2FoldChange, y = padj, color = (padj<0.01 & abs(log2FoldChange) > 1
  facet_wrap(result_name ~ .) +
  geom_point()
```



```r
# Histogram for log fold change vs padj
ggplot(filtered_results, aes(x = log2FoldChange, fill = result_name)) +
  facet_wrap(result_name ~ .) +
  geom_histogram(bins = 50)
```



IGV Results



This are the timepoint tracks for gene H19 using the GRCm38/mm10 genome

Future directions

Pretty cool video on DESeq2 analysis: https://www.youtube.com/watch?v=NGbZmlGLG5w&t=264s

Ideas of plots to include: - Circos plot - Needs to know where the gene is located so need to use biomaRt to get data from ensembl and get the gene information (it looks a bit complicated) - Will need to save a csv file with Gene ID, chromosome name, start position, end position + rows in results from DESeq2 - Plots can be made in Circa (easier) or through bash (harder)