Pre-processing of *Strongyloides stercoralis* bulk RNAseq via an alignment-free analysis pipeline

Contents

1 Introduction

The human parasitic nematode *Strongyloides stercoralis* is estimated to infect approximately 610 million people (Buonfrate *et al* 2020). Similar to other intestinal parasitic nematodes, *S. stercoralis* has a complex life cycle in which distinct developmental stages navigate starkly different environments via life-stage specific behavioral preferences (Castelletto *et al* 2014, Bryant and Hallem 2018a). Across nematode species, ethological and behavioral differences are often reflected in the temporal regulation of gene expression. For *S. stercoralis*, several studies have utilized bulk RNA sequencing to probe the genomic basis of parasitism by identifying gene families that are uniquely upregulated in parasitic life stages (Stolzfus *et al* 2012, Hunt *et al* 2016, Hunt *et al* 2018). These efforts highlight a feature of modern bioinformatics - the secondary analysis of publically avaliable datasets. Here, an original dataset featuring bulk RNA sequencing of seven *S. stercoralis* developmental stages was initially aligned to genomic contigs (6 December 2011 draft, Stolzfus *et al* 2012). Subsequently, a subset of this dataset was re-analyzed coincident with the release of a high-quality draft assembly (Hunt *et al* 2016); this re-analysis focused on the differences between three life stages: free-living adults that navigate the environment, parasitic adults located within the host intestine, and infective third-stage larvae that actively engage in host-seeking behaviors. As genome assembly and RNA sequencing of additional parasitic nematode species continues, the publically-avaliable *S. stercoralis* RNAseq dataset continues to be utilized for cross-species and cross-life stage comparisons (Hunt *et al* 2016, Hunt *et al* 2018). Furthermore, several helminth RNA-seq datasets, including *S. stercoralis* were realigned to their reference genomes and integrated into WormBase Parasite, where they are accessible to the field at large in the form of genome-aligned RNA-seq expression tracks (Howe *et al* 2017).

As research into the genomic basis of parasitism in helminths continues, access to quantitative gene expression levels will greatly enhance studies into the functional role of specific genes and gene families. Differential gene expression results have been published as supplemental data (Hunt *et al* 2016), however these analyses only included pairwise comparisons between three life stages. Quantitative comparisons that utilize the full seven avaliable life stages have not yet been published.

2 Pre-processing Methods Overview

The introduction of alignment-free read mapping tools such as Kallisto has significantly lowered the computational barrier for re-analysis of publically avaliable RNAseq datasets. Here, I take advantage of these tools to re-analyze the *S. sterocoralis* bulk RNA-seq data, extending differential gene expression analyses across all avaliable developmental stages. Kallisto and custom R scripts were used to perform ultra-fast read mapping of raw reads to the *S. stercoalis* reference transcriptome (PRJEB528.WBPS14.mRNA_transcripts, downloaded from WormBase Parasite on 16 June 2020).

Kallisto alignments are imported into the R environment and annotated with information imported via the Wormbase ParaSite BioMaRT. Annotation information includes: *C. elegans* homologs/percent homology, UniProtKB number, Interpro terms, GO terms, and general Description information. Hunt *et al* 2016 establishes two distinct subclades from the four sequenced *Strongyloides* species: *S. venezuelensis-S. papillosus* and *S. ratti-S. stercoralis*. Thus, we also include annotation information for the appropriate in-group (here, *S. ratti*), and a reference member of the out-group (*S. papillosus*). Annotation information can be saved as an R object that is passed to a Shiny Web App for downstream browsing and on-demand analysis. Note: raw count data could be saved as a digitial gene expression list if desired (not currently done).

Raw reads were quantified as counts per million using the EdgeR package, then filtered to remove transcripts with low counts (less than 1 count-per-million in at least 3 samples). A list of discarded genes and their expression values across life stages can be saved. Non-discarded gene values are then normalized using the trimmed mean of M-values method (TMM, Robinson and Oshlack) to permit between-samples comparisons. The mean-variance relationship was modeled using a precision weights approach Law *et al* 2014.

A variance-stabilized DGEList object can be saved; this file is passed to a Shiny Web App for downstream browsing and on-demand analysis.

3 Results/Analysis

Note: Code chunks are collated and echoed at the end of the document in the Appendix.

3.1 Kallisto read mapping

This shell script checks the qualitiy of the fastq files and performs an alignment to the Strongyloides stercoralis cDNA transcriptome reference with Kallisto; to work, it needs to be saved as an .sh file in a folder containing required raw files.This script is reproduced here to demonstrate the QC and alignment process.

3.2 Import Kallisto reads into R

Import Kallisto transcript counts into R using Tximport. Counts are generated from abundance files using the `lengthScaledTPM` option. This code chunk is not evaluated each time, instead it was run once and an object containing the transcripts per million data is saved. In subsequent chunks, that file is loaded, and analysis progresses. The point of this is so that folks attempting to rerun this analysis do not need to have abundance files loaded on their local machines (and we do not have to upload abundance files to github).

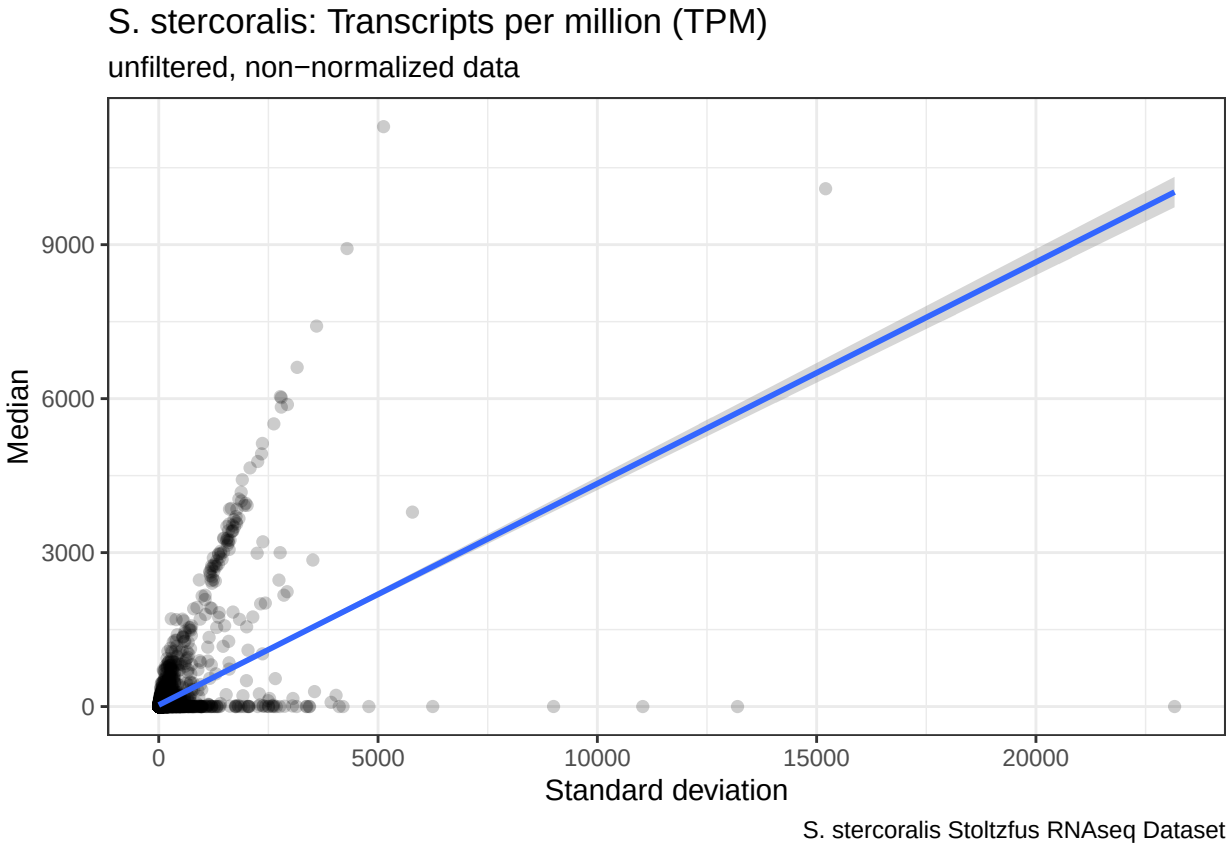
3.3 Gene Annotation

Import gene annotation information for *S. stercoralis* genes, including:
* *C. elegans* homologs/percent homology

* *S. ratti* homologs/percent homology
* *S. papillosus* homologs/percent homology
* UniProtKB number
* Interpro terms
* GO terms
* general Description information using biomart.

3.4 Generate Digital Gene Expression List

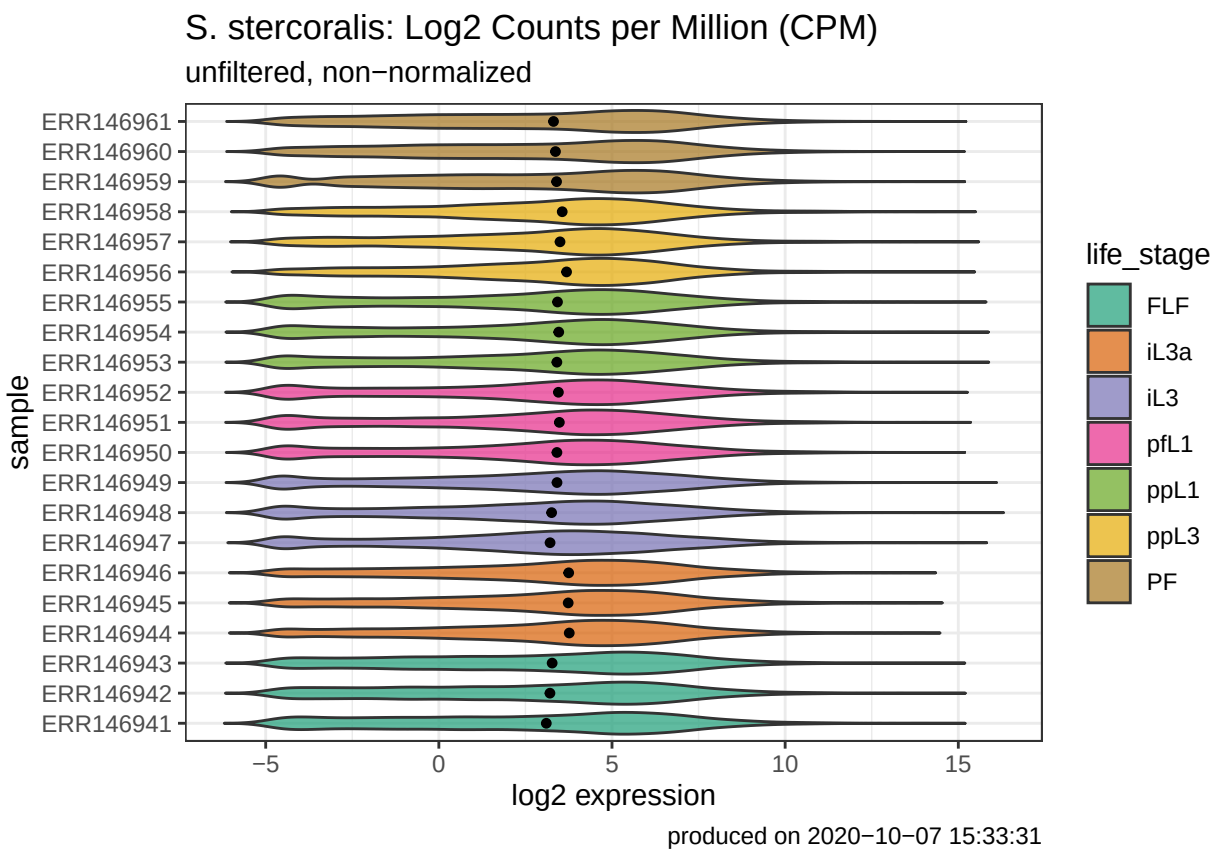
This chunk of code generates a digitial gene expression list that could be easily shared/loaded for downstream filtering/normalization. It generates a scatter plot of unfiltered and non-normalized transcripts per million data.



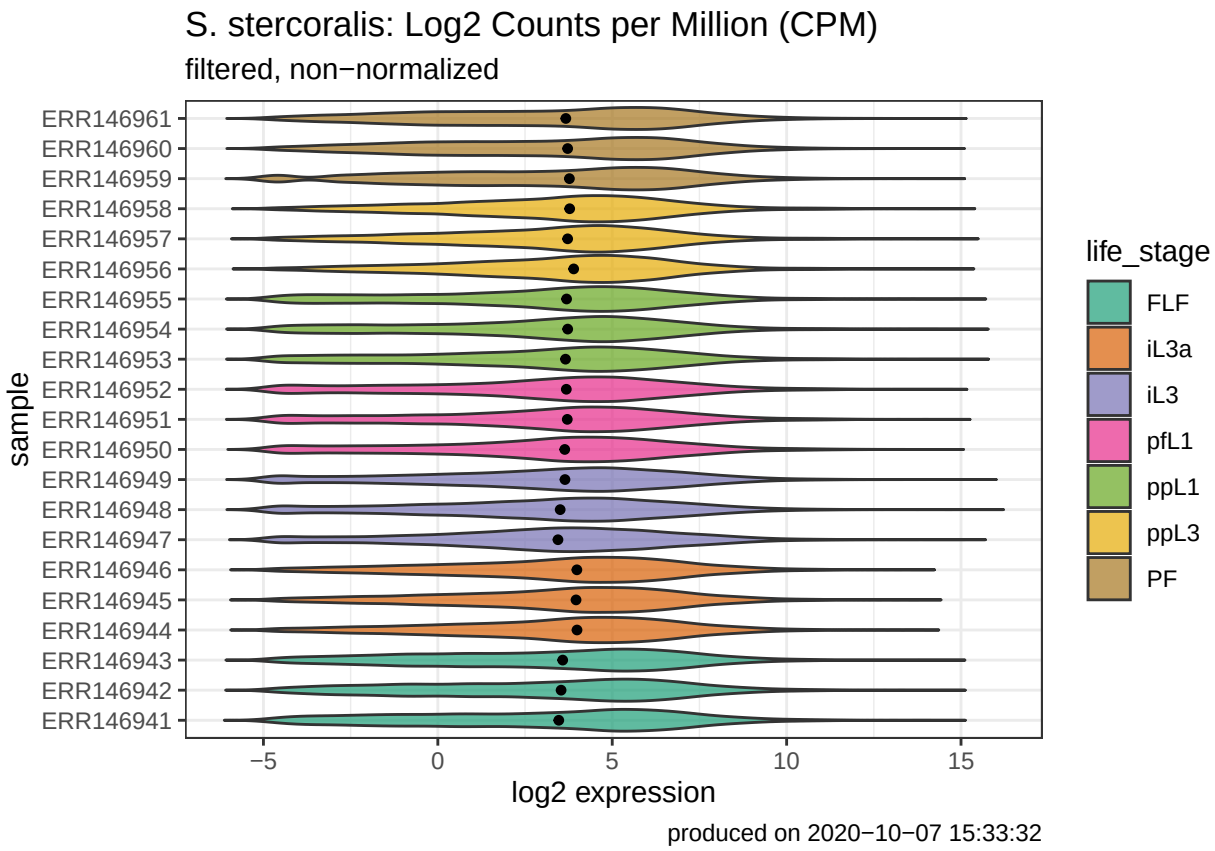
3.5 Data Filtering and Normalization

The goal of this chunk is to:
1. Filter and normalize data
2. Use `ggplot2` to visualize the impact of filtering and normalization on the data.

3.5.1 Plot of unfiltered, non-normalized log2CPM data by life stage

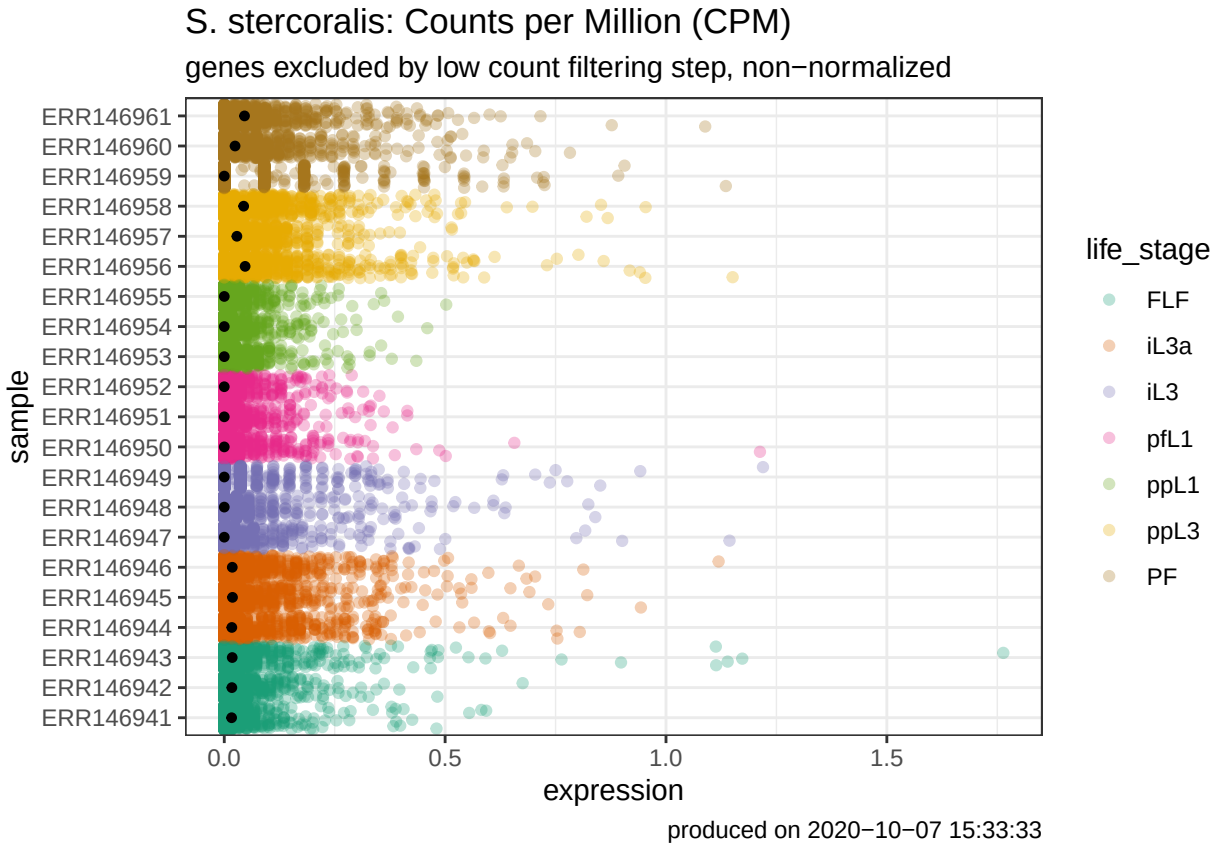


3.5.2 Plot of filtered, non-normalized log2CPM data by life stage

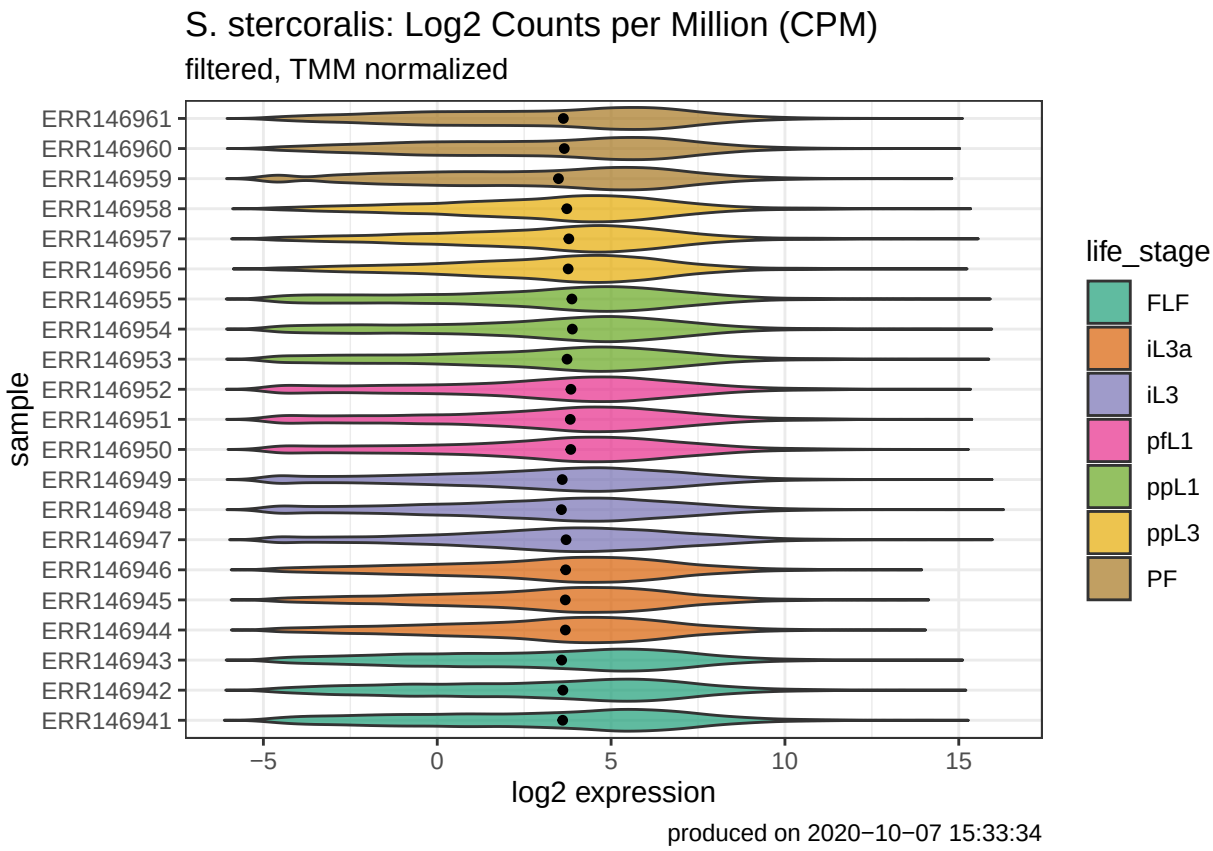


3.5.3 Plot of genes discarded by low-copy filtering step

The low copy number filtering step excluded a total of `dim(myDGEList.discarded)[[1]]` genes.

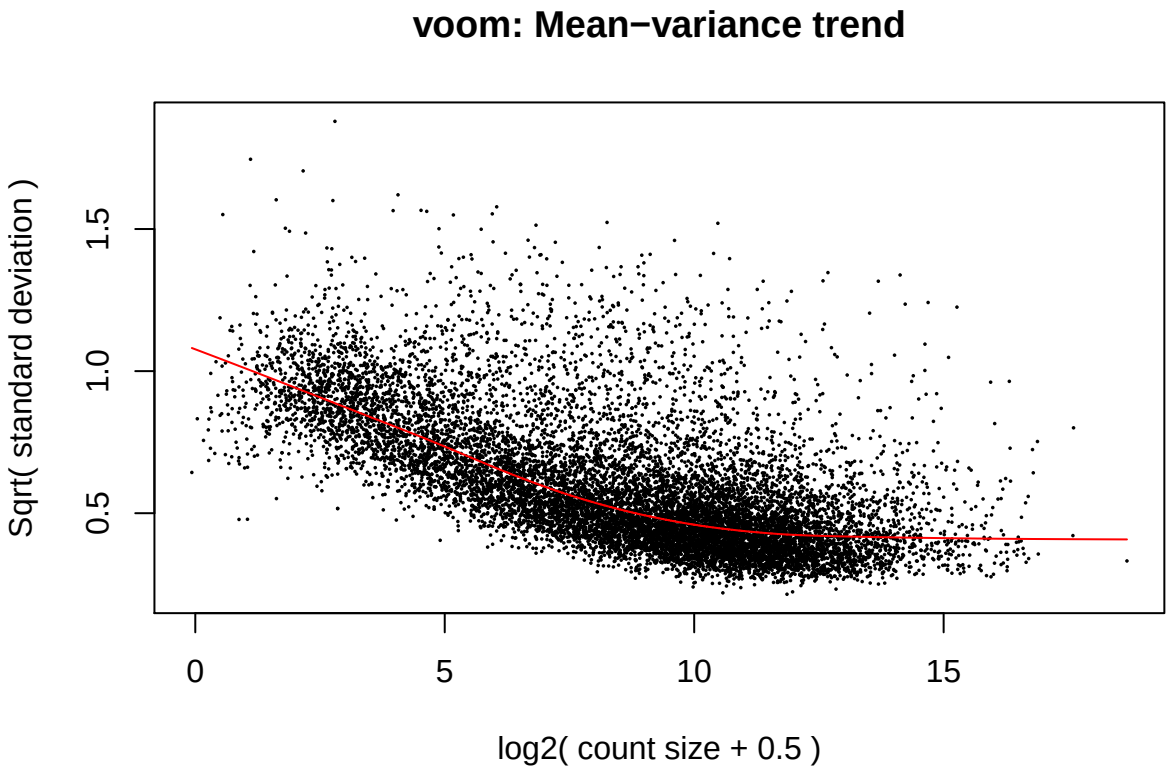

S. stercoralis: Counts per Million (CPM)
genes excluded by low count filtering step, non−normalized
ERR146961
ERR146960
ERR146959
ERR146958
ERR146957
ERR146956
ERR146955
ERR146954
ERR146953
ERR146952
ERR146951
ERR146950
ERR146949
ERR146948
ERR146947
ERR146946
ERR146945
ERR146944
ERR146943
ERR146942
ERR146941
sample
0.0
0.5
1.0
1.5
expression
life_stage
FLF
iL3a
iL3
pfL1
ppL1
ppL3
PF
produced on 2020−10−07 15:33:33

3.5.4 Plot of filtered, normalized log2CPM data by life stage



3.6 Compute Variance-Stabilized DGEList Object

This chunk uses a DGEList of filtered and normalized abundance data. It will fit data to a linear model for responsively detecting differentially expressed genes (DEGs).



3.7 Save Data and Annotations

This code chunk saves data and annotations, generated in code chunks above. The filtered data and annotation information is required for downstream analyses. It enables users to not have to re-import and re-align raw read files every time the code is run. The variance-stabilized vDGEList can be imported into a Shiny data browsing/analysis app.

4 Appendix I : All code for this report

```
knitr::opts_chunk$set(echo = FALSE, message = FALSE, warning = FALSE)

# This script checks the qualitiy of the fastq files and performs
# an alignment to the Strongyloides stercoralis cDNA transcriptome
# reference with Kallisto.
# To run this 'shell script' you will need to open your terminal and
# navigate to the directory where this script resides on your computer.
# This should be the same directory where you fastq files and reference
# fasta file are found.
# Change permissions on your computer so that you can run a
# shell script by typing: 'chmod u+x readMapping.sh' (without the quotes)
# at the terminal prompt
# Then type './readMapping.sh' (without the quotes) at the prompt.
# This will begin the process of running each line of code in the shell script.

# first use fastqc to check the quality of our fastq files:
```

```
fastqc *.gz -t 14

# next, we want to build an index from our reference fasta file
# I got my reference Strongyloides sterocralis transcripts
# from WormBase Parasite
kallisto index -i strongyloides_stercoralis.PRJEB528.WBPS14.mRNA_transcripts.index
strongyloides_stercoralis.PRJEB528.WBPS14.mRNA_transcripts.fa

# now map reads to the indexed reference host transcriptome
# use as many 'threads' as your machine will allow in order
# to speed up the read mapping process.
# note that we're also including the '&>' at the end of each line
# this takes the information that would've been printed to our
# terminal, and outputs this in a log file that is saved in /data/course_data

# Free-Living Females
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS14.mRNA_transcripts.index -o
ERR146941 -t 14 ERR146941_1.fastq.gz ERR146941_2.fastq.gz&> ERR146941.log
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS14.mRNA_transcripts.index -o
ERR146942 -t 14 ERR146942_1.fastq.gz ERR146942_2.fastq.gz&> ERR146942.log
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS14.mRNA_transcripts.index -o
ERR146943 -t 14 ERR146943_1.fastq.gz ERR146943_2.fastq.gz&> ERR146943.log

# L3i+ (Activated iL3s)
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS14.mRNA_transcripts.index -o
ERR146944 -t 14 ERR146944_1.fastq.gz ERR146944_2.fastq.gz&> ERR146944.log
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS14.mRNA_transcripts.index -o
ERR146945 -t 14 ERR146945_1.fastq.gz ERR146945_2.fastq.gz&> ERR146945.log
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS14.mRNA_transcripts.index -o
ERR146946 -t 14 ERR146946_1.fastq.gz ERR146946_2.fastq.gz&> ERR146946.log

# L3i
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS14.mRNA_transcripts.index -o
ERR146947 -t 14 ERR146947_1.fastq.gz ERR146947_2.fastq.gz&> ERR146947.log
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS14.mRNA_transcripts.index -o
ERR146948 -t 14 ERR146948_1.fastq.gz ERR146948_2.fastq.gz&> ERR146948.log
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS14.mRNA_transcripts.index -o
ERR146949 -t 14 ERR146949_1.fastq.gz ERR146949_2.fastq.gz&> ERR146949.log

# Post-Free-Living L1s
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS14.mRNA_transcripts.index -o
ERR146950 -t 14 ERR146950_1.fastq.gz ERR146950_2.fastq.gz&> ERR146950.log
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS14.mRNA_transcripts.index -o
ERR146951 -t 14 ERR146951_1.fastq.gz ERR146951_2.fastq.gz&> ERR146951.log
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS14.mRNA_transcripts.index -o
ERR146952 -t 14 ERR146952_1.fastq.gz ERR146952_2.fastq.gz&> ERR146952.log

# Post-Parasitic L1s
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS14.mRNA_transcripts.index -o
ERR146953 -t 14 ERR146953_1.fastq.gz ERR146953_2.fastq.gz&> ERR146953.log
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS14.mRNA_transcripts.index -o
ERR146954 -t 14 ERR146954_1.fastq.gz ERR146954_2.fastq.gz&> ERR146954.log
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS14.mRNA_transcripts.index -o
```

```
ERR146955 -t 14 ERR146955_1.fastq.gz ERR146955_2.fastq.gz&> ERR146955.log

# Post-Parasitic L1s
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS14.mRNA_transcripts.index -o
ERR146956 -t 14 ERR146956_1.fastq.gz ERR146956_2.fastq.gz&> ERR146956.log
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS14.mRNA_transcripts.index -o
ERR146957 -t 14 ERR146957_1.fastq.gz ERR146957_2.fastq.gz&> ERR146957.log
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS14.mRNA_transcripts.index -o
ERR146958 -t 14 ERR146958_1.fastq.gz ERR146958_2.fastq.gz&> ERR146958.log

# Parasitic Females
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS14.mRNA_transcripts.index -o
ERR146959 -t 14 ERR146959_1.fastq.gz ERR146959_2.fastq.gz&> ERR146959.log
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS14.mRNA_transcripts.index -o
ERR146960 -t 14 ERR146960_1.fastq.gz ERR146960_2.fastq.gz&> ERR146960.log
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS14.mRNA_transcripts.index -o
ERR146961 -t 14 ERR146961_1.fastq.gz ERR146961_2.fastq.gz&> ERR146961.log



# summarize fastqc and kallisto mapping results
# in a single summary html using MultiQC
multiqc -d .

echo "Finished"
# load packages ----
suppressPackageStartupMessages({
  library(tidyverse)
  library(tximport)
  library(ensembldb)
  library(biomaRt)
  library(magrittr)
})
# read in the study design ----
targets <- read_tsv("../Data/S_stercoralis/Study_Design/PRJEB3116_study_design.txt",
                    na = c("", "NA", "na"))
# create file paths to the abundance files generated by Kallisto
# using the 'file.path' function
path <- file.path("../Data/S_stercoralis/Reads",
                  targets$sample,
                  "abundance.tsv")

# get annotations using organism-specific package ----
Tx.Ss <- getBM(attributes=c('wbps_transcript_id',
                            'wbps_gene_id'),
               # grab the ensembl annotations for Wormbase Parasite genes
               mart = useMart(biomart="parasite_mart",
                              dataset = "wbps_gene",
                              host="https://parasite.wormbase.org",
                              port = 443),
               filters = c('species_id_1010'),
               value = list('ststerprjeb528')) %>%
  as_tibble() %>%
```

```
  #we need to rename the columns retreived from biomart
  dplyr::rename(target_id = wbps_transcript_id,
                gene_name = wbps_gene_id)


# import Kallisto transcript counts into R using Tximport ----
# copy the abundance files to the working directory and
# rename so that each sample has a unique name
Txi_gene <- tximport(path,
                     type = "kallisto",
                     tx2gene = Tx.Ss[,1:2],
                     txOut = FALSE,
                     countsFromAbundance = "lengthScaledTPM",
                     ignoreTxVersion = FALSE)

# Save the raw transcript counts ----
save(Txi_gene,
     file = file.path("../Data/S_stercoralis",
                      "SsRNAseq_TPM"))
# Introduction to this chunk -----------
# This chunk imports gene annotation information
# for S. stercoralis genes, including:
# C. elegans homologs/percent homology,
# S. ratti homologs/percent homology,
# S. papillosus homologs/percent homology,
# UniProtKB number, Interpro terms, GO terms,
# and general Description information using biomart.
# It will generate a table that can be saved.

# Load packages ------
suppressPackageStartupMessages({
  library(tidyverse)
  library(tximport)
  library(ensembldb)
  library(biomaRt)
  library(magrittr)
})

# Load data & study design ----
load(file = file.path("../Data/S_stercoralis",
                      "SsRNAseq_TPM"))

targets <- read_tsv("../Data/S_stercoralis/Study_Design/PRJEB3116_study_design.txt",
                    na = c("", "NA", "na"))

# Get In-subclade group homologs for S. stercoralis
# genes from BioMart and filter -----
Annt.temp.1 <- getBM(attributes=c('wbps_gene_id',
                                  'strattprjeb125_gene_name',
                                  'strattprjeb125_homolog_perc_id'),
                     # grab the ensembl annotations for Wormbase Parasite genes
                     mart = useMart(biomart="parasite_mart",
                                    dataset = "wbps_gene",
```

```
                                    host="https://parasite.wormbase.org",
                                    port = 443),
                     filters = c('species_id_1010'),
                     value = list('stsstерprjeb528')) %>%
  as_tibble() %>%
  #rename columns
  dplyr::rename(geneID = wbps_gene_id,
                In.subclade_geneID = strattprjeb125_gene_name,
                In.subclade_percent_homology= strattprjeb125_homolog_perc_id
  ) %>%
  dplyr::group_by(geneID)

# Get Out-subclade group homologs for S. stercoralis
# genes from BioMart and filter -----
Annt.temp.2 <- getBM(attributes=c('wbps_gene_id',
                                  'stpapiprjeb525_gene',
                                  'stpapiprjeb525_homolog_perc_id'
                                  ),
                     # grab the ensembl annotations for Wormbase Parasite genes
                     mart = useMart(biomart="parasite_mart",
                                    dataset = "wbps_gene",
                                    host="https://parasite.wormbase.org",
                                    port = 443),
                     filters = c('species_id_1010'),
                     value = list('stsstерprjeb528')) %>%
  as_tibble() %>%
  #rename columns
  dplyr::rename(geneID = wbps_gene_id,
                Out.subclade_geneID = stpapiprjeb525_gene,
                Out.subclade_percent_homology= stpapiprjeb525_homolog_perc_id
  ) %>%
  dplyr::group_by(geneID)

# Get C. elegans homologs and gene information for S. stercoralis
# genes from BioMart and filter -----
Annt.temp.3 <- getBM(attributes=c('wbps_gene_id',
                                  'caelegprjna13758_gene_name',
                                  'caelegprjna13758_homolog_perc_id',
                                  'description',
                                  'interpro_short_description',
                                  'go_name_1006',
                                  'uniprot_sptrembl'),
                     # grab the ensembl annotations for Wormbase Parasite genes
                     mart = useMart(biomart="parasite_mart",
                                    dataset = "wbps_gene",
                                    host="https://parasite.wormbase.org",
                                    port = 443),
                     filters = c('species_id_1010'),
                     value = list('stsstерprjeb528')) %>%
  as_tibble() %>%
  #rename columns
  dplyr::rename(geneID = wbps_gene_id,
                Ce_geneID = caelegprjna13758_gene_name,
```

```
                Ce_percent_homology = caelegprjna13758_homolog_perc_id,
                Description = description,
                GO_term = go_name_1006,
                UniProtKB = uniprot_sptrembl
  ) %>%
  dplyr::group_by(geneID)

Annt.import <- full_join(Annt.temp.1, Annt.temp.2, by = "geneID") %>%
  full_join(Annt.temp.3, by = "geneID")

# Replace empty string values (mostly in Ce_geneID column) with NAs
Annt.import[Annt.import == ""]<-NA

# Remove any duplications in the possible homolog matches.
# Select based on highest % homology.
# Give fake value here to make sure genes
# without homologs aren't filtered out
Annt.import$Ce_percent_homology[
  is.na(Annt.import$Ce_percent_homology)] <- 1000
Annt.import$In.subclade_percent_homology[
  is.na(Annt.import$In.subclade_percent_homology)] <- 1000
Annt.import$Out.subclade_percent_homology[
  is.na(Annt.import$Out.subclade_percent_homology)] <- 1000


Annt.logs <-Annt.import %>%
  dplyr::select(!c(interpro_short_description:GO_term))%>%
  group_by(geneID) %>%
  slice_max(n = 1, order_by = Ce_percent_homology,
            with_ties = FALSE) %>%
  slice_max(n = 1, order_by = In.subclade_percent_homology,
            with_ties = FALSE) %>%
  slice_max(n = 1, order_by = Out.subclade_percent_homology,
            with_ties = FALSE) %>%
  group_by(geneID, Ce_geneID)

# Remove source code to shorten the description
Annt.logs$Description<- Annt.logs$Description %>%
  str_replace_all(string = .,
                  pattern = "  \\[Source:.*\\]",
                  replacement = "") %>%
  cbind()

Annt.logs$Ce_percent_homology[
  Annt.logs$Ce_percent_homology == 1000] <- NA
Annt.logs$In.subclade_percent_homology[
  Annt.logs$In.subclade_percent_homology == 1000]<- NA
Annt.logs$Out.subclade_percent_homology[
  Annt.logs$Out.subclade_percent_homology == 1000]<- NA

# Clean up interprotKB terms, removing duplications and collapsing to one line
Annt.interpro<-Annt.import %>%
  dplyr::select(geneID, Ce_geneID, interpro_short_description) %>%
```

```
  group_by(geneID, Ce_geneID) %>%
  dplyr::distinct(interpro_short_description, .keep_all = TRUE) %>%
  dplyr::summarise(InterPro = paste(interpro_short_description,
                                    collapse = ', '))
# Clean up GO terms, removing duplications and collapsing to one line
Annt.goterms<-Annt.import %>%
  dplyr::select(geneID, Ce_geneID, GO_term) %>%
  group_by(geneID, Ce_geneID) %>%
  dplyr::distinct(GO_term, .keep_all = TRUE) %>%
  dplyr::summarise(GO_term = paste(GO_term, collapse = ', '))

annotations<-dplyr::left_join(Annt.logs, Annt.interpro) %>%
  dplyr::left_join(.,Annt.goterms) %>%
  ungroup() %>%
  dplyr::relocate(In.subclade_geneID,
                  In.subclade_percent_homology,
                  Out.subclade_geneID,
                  Out.subclade_percent_homology,
                  .after = geneID) %>%
  column_to_rownames(var = "geneID")

# Goals of this chunk:
# Generate a digitial gene expression list
# that could be easily shared/loaded for downstream filtering/normalization

# Load packages -----
suppressPackageStartupMessages({
  library(tidyverse)
  library(edgeR)
  library(matrixStats)
  library(cowplot)
  library(ggthemes)
  library(RColorBrewer)
  library(gprofiler2)
})

# Generate and plot summary stats for the data ----
myTPM.stats <- transform(Txi_gene$abundance,
                         SD=rowSds(Txi_gene$abundance),
                         AVG=rowMeans(Txi_gene$abundance),
                         MED=rowMedians(Txi_gene$abundance))

# produce a scatter plot of the transformed data
p1<-ggplot(myTPM.stats) +
  aes(x = SD, y = MED) +
  geom_point(shape=16, size=2, alpha = 0.2) +
  geom_smooth(method=lm) +
  #geom_hex(show.legend = FALSE) +
  labs(y="Median", x = "Standard deviation",
       title = "S. stercoralis: Transcripts per million (TPM)",
       subtitle="unfiltered, non-normalized data",
       caption="S. stercoralis Stoltzfus RNAseq Dataset") +
  theme_bw()
```

```
p1

# make a Digital Gene Expression list using the raw counts and plot ----
myDGEList <- DGEList(Txi_gene$counts,
                     samples = targets$sample,
                     group = targets$group,
                     genes = annotations)

# Goals of this chunk:
# 1 - Filter and normalize data
# 2 - use ggplot2 to visualize the impact of filtering and
# normalization on the data.

# Notes:
# recall that abundance data are TPM, while the counts are
# read counts mapping to each gene or transcript

# Load packages -----
suppressPackageStartupMessages({
  library(tidyverse)
  library(edgeR)
  library(matrixStats)
  library(cowplot)
  library(ggthemes)
  library(RColorBrewer)
  library(gprofiler2)
})

# calculate and plot log2 counts per million ----
# Generate life stage IDs
ids <- rep(cbind(targets$group),
           times = nrow(myDGEList$counts)) %>%
  as_factor()

# use the 'cpm' function from EdgeR to get log2 counts per million
# then coerce into a tibble
# add sample names to the dataframe
# tidy up the dataframe into a tibble
log2.cpm.df.pivot <-cpm(myDGEList, log=TRUE) %>%
  as_tibble(rownames = "geneID") %>%
  setNames(nm = c("geneID", targets$sample)) %>%
  pivot_longer(cols = -geneID,
               names_to = "samples",
               values_to = "expression") %>%
  add_column(life_stage = ids)


# plot the pivoted data
p2 <- ggplot(log2.cpm.df.pivot) +
  aes(x=samples, y=expression, fill=life_stage) +
  geom_violin(trim = FALSE, show.legend = T, alpha= 0.7) +
  stat_summary(fun = "median",
               geom = "point",
```

```
               shape = 20,
               size = 2,
               color = "black",
               show.legend = FALSE) +
  labs(y="log2 expression", x = "sample",
       title = "S. stercoralis: Log2 Counts per Million (CPM)",
       subtitle="unfiltered, non-normalized",
       caption=paste0("produced on ", Sys.time())) +
  theme_bw() +
  scale_fill_brewer(palette = "Dark2") +
  coord_flip()

p2

# Filter the data ----

# filter genes/transcripts with low counts
# how many genes had more than 1 CPM (TRUE) in at least n samples
# Note: The cutoff "n" is adjusted for the number of
# samples in the smallest group of comparison.
keepers <- cpm(myDGEList) %>%
  rowSums(.>1)>=3

myDGEList.filtered <- myDGEList[keepers,]

ids.filtered <- rep(cbind(targets$group),
                    times = nrow(myDGEList.filtered)) %>%
  as_factor()

log2.cpm.filtered.df.pivot <- cpm(myDGEList.filtered, log=TRUE) %>%
  as_tibble(rownames = "geneID") %>%
  setNames(nm = c("geneID", targets$sample)) %>%
  pivot_longer(cols = -geneID,
               names_to = "samples",
               values_to = "expression") %>%
  add_column(life_stage = ids.filtered)

p3 <- ggplot(log2.cpm.filtered.df.pivot) +
  aes(x=samples, y=expression, fill=life_stage) +
  geom_violin(trim = FALSE, show.legend = T, alpha= 0.7) +
  stat_summary(fun = "median",
               geom = "point",
               shape = 20,
               size = 2,
               color = "black",
               show.legend = FALSE) +
  labs(y="log2 expression", x = "sample",
       title = "S. stercoralis: Log2 Counts per Million (CPM)",
       subtitle="filtered, non-normalized",
       caption=paste0("produced on ", Sys.time())) +
  theme_bw() +
  scale_fill_brewer(palette = "Dark2") +
  coord_flip()
```

```
p3

# Look at the genes excluded by the filtering step ----
# just to check that there aren't any with
# high expression that are in few samples
# Discarded genes
myDGEList.discarded <- myDGEList[!keepers,]

ids.discarded <- rep(cbind(targets$group),
                     times = nrow(myDGEList.discarded)) %>%
  as_factor()

log2.cpm.discarded.df.pivot <- cpm(myDGEList.discarded, log=F) %>%
  as_tibble(rownames = "geneID") %>%
  setNames(nm = c("geneID", targets$sample)) %>%
  pivot_longer(cols = -geneID,
               names_to = "samples",
               values_to = "expression") %>%
  add_column(life_stage = ids.discarded)

p.discarded <- ggplot(log2.cpm.discarded.df.pivot) +
  aes(x=samples, y=expression, color=life_stage) +
  geom_jitter(alpha = 0.3, show.legend = T)+
  stat_summary(fun = "median",
               geom = "point",
               shape = 20,
               size = 2,
               color = "black",
               show.legend = FALSE) +
  labs(y="expression", x = "sample",
       title = "S. stercoralis: Counts per Million (CPM)",
       subtitle="genes excluded by low count filtering step, non-normalized",
       caption=paste0("produced on ", Sys.time())) +
  theme_bw() +
  scale_color_brewer(palette = "Dark2") +
  coord_flip()

p.discarded

# # Carry out GO enrichment of discarded gene set using gProfiler2 ----
# discarded.geneID <- unique(log2.cpm.discarded.df.pivot$geneID)
# gost.res <- gost(list(Discarded_genes = discarded.geneID),
#                  organism = "ststerprjeb528", correction_method = "fdr")
# gostplot(gost.res, interactive = T, capped = T)

# Genes that are above 1 cpm
log2.cpm.discarded.df.pivot %>%
  dplyr::filter(expression > 1)

# Generate a matrix of discarded genes and their raw counts ----
discarded.gene.df <- log2.cpm.discarded.df.pivot %>%
  pivot_wider(names_from = c(life_stage, samples),
              names_sep = "-",
```

```
              values_from = expression,
              id_cols = geneID)

# Normalize the data using a between samples normalization ----
# Source for TMM sample normalization here:
# https://genomebiology.biomedcentral.com/articles/10.1186/gb-2010-11-3-r25
myDGEList.filtered.norm <- calcNormFactors(myDGEList.filtered, method = "TMM")

log2.cpm.filtered.norm <- cpm(myDGEList.filtered.norm, log=TRUE)

log2.cpm.filtered.norm.df<- cpm(myDGEList.filtered.norm, log=TRUE) %>%
  as_tibble(rownames = "geneID") %>%
  setNames(nm = c("geneID", targets$sample))

log2.cpm.filtered.norm.df.pivot<-log2.cpm.filtered.norm.df %>%
  pivot_longer(cols = -geneID,
               names_to = "samples",
               values_to = "expression") %>%
  add_column(life_stage = ids.filtered)

p4 <- ggplot(log2.cpm.filtered.norm.df.pivot) +
  aes(x=samples, y=expression, fill=life_stage) +
  geom_violin(trim = FALSE, show.legend = T, alpha = 0.7) +
  stat_summary(fun = "median",
               geom = "point",
               shape = 20,
               size = 2,
               color = "black",
               show.legend = FALSE) +
  labs(y="log2 expression", x = "sample",
       title = "S. stercoralis: Log2 Counts per Million (CPM)",
       subtitle="filtered, TMM normalized",
       caption=paste0("produced on ", Sys.time())) +
  theme_bw() +
  scale_fill_brewer(palette = "Dark2") +
  coord_flip()

p4

# Introduction to this chunk ----
# This chunk uses a DGEList of filtered and normalized abundance data
# It will fit data to a linear model for responsively detecting
# differentially expressed genes (DEGs)

# Load packages ----
suppressPackageStartupMessages({
  library(tidyverse)
  library(limma) # differential gene expression using linear modeling
  library(edgeR)
})

# Set up the design matrix ----
# no intercept/blocking for matrix, comparisons across group
```

```
group <- factor(targets$group)
design <- model.matrix(~0 + group)
colnames(design) <- levels(group)

# NOTE: To handle a 'blocking' design' or a batch effect, use:
# design <- model.matrix(~block + treatment)

# Model mean-variance trend and fit linear model to data ----
# Use VOOM function from Limma package to model the mean-variance relationship
# produces a variance-stabilized DEGList, that include precision
# weights for each gene to try and control for heteroscedasity.
# transforms count data to log2-counts per million
# Outputs: E = normalized expression values on the log2 scale
v.DEGList.filtered.norm <- voom(counts = myDGEList.filtered.norm,
                                design = design, plot = T)
colnames(v.DEGList.filtered.norm)<-targets$sample
colnames(v.DEGList.filtered.norm$E) <- paste(targets$group,
                                             targets$sample,sep = '-')

# Check for presence of output folder, generate if it doesn't exist
output.path <- "../Outputs"
if (!dir.exists(output.path)){
  dir.create(output.path)
}

# Save full gene annotations ----
save(annotations,
file = file.path(output.path,
                         "Ss_geneAnnotations"))

# Save DGEList of raw counts ----
save(myDGEList,
     file = file.path(output.path,
                              "SsRNAseq_DGEList"))

# Save a matrix of discarded genes and their raw counts ----
discarded.gene.df %>%
write.csv(file = file.path(output.path,
                           "SsRNAseq_discardedGene_counts.csv"))

# This data is required for downstream analyses in this file.
# It enables users to not have to re-import and re-align
# raw read files every time the code is run.

SsRNAseq.preprocessed.data <- list(targets = targets,
                                   annotations = annotations,
                                   log2.cpm.filtered.norm = log2.cpm.filtered.norm,
                                   myDGEList.filtered.norm = myDGEList.filtered.norm
)
save(SsRNAseq.preprocessed.data,
     file = file.path(output.path,
                      "SsRNAseq_data_preprocessed"))
```

```r
# Save matrix of genes and their filtered, normalized counts ----
colnames(log2.cpm.filtered.norm)<-paste(targets$group,
                                        targets$sample,
                                        sep = "-")
write.csv(log2.cpm.filtered.norm,
          file = file.path(output.path,
                           "SsRNAseq_log2cpm_filtered_norm.csv"))

# Save v.DEGList ----
# This file will be imported into Shiny App

save(v.DEGList.filtered.norm,
     file = file.path(output.path,
                      "Ss_vDGEList")
)

sessionInfo()
```

5 Appendix II: Session Info

```
## R version 3.6.3 (2020-02-29)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS Catalina 10.15.5
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats4    parallel  stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] gprofiler2_0.1.9        RColorBrewer_1.1-2      ggthemes_4.2.0
##  [4] cowplot_1.0.0           matrixStats_0.56.0      edgeR_3.28.1
##  [7] limma_3.42.2            magrittr_1.5            biomaRt_2.42.1
## [10] ensembldb_2.10.2        AnnotationFilter_1.10.0 GenomicFeatures_1.38.2
## [13] AnnotationDbi_1.48.0    Biobase_2.46.0          GenomicRanges_1.38.0
## [16] GenomeInfoDb_1.22.1     IRanges_2.20.2          S4Vectors_0.24.4
## [19] BiocGenerics_0.32.0     tximport_1.14.2         forcats_0.5.0
## [22] stringr_1.4.0           dplyr_1.0.1             purrr_0.3.4
## [25] readr_1.3.1             tidyr_1.1.1             tibble_3.0.3
## [28] ggplot2_3.3.2           tidyverse_1.3.0
##
## loaded via a namespace (and not attached):
##  [1] colorspace_1.4-1            ellipsis_0.3.1
##  [3] XVector_0.26.0              fs_1.4.2
##  [5] rstudioapi_0.11             farver_2.0.3
##  [7] bit64_0.9-7                 fansi_0.4.1
##  [9] lubridate_1.7.9             xml2_1.3.2
```

```
## [11] splines_3.6.3               knitr_1.29
## [13] jsonlite_1.7.0              Rsamtools_2.2.3
## [15] broom_0.5.6                 dbplyr_1.4.4
## [17] compiler_3.6.3              httr_1.4.2
## [19] backports_1.1.8             assertthat_0.2.1
## [21] Matrix_1.2-18               lazyeval_0.2.2
## [23] cli_2.0.2                   htmltools_0.5.0
## [25] prettyunits_1.1.1           tools_3.6.3
## [27] gtable_0.3.0                glue_1.4.1
## [29] GenomeInfoDbData_1.2.2      rappdirs_0.3.1
## [31] Rcpp_1.0.5                  cellranger_1.1.0
## [33] vctrs_0.3.2                 Biostrings_2.54.0
## [35] nlme_3.1-148                rtracklayer_1.46.0
## [37] xfun_0.15                   rvest_0.3.5
## [39] lifecycle_0.2.0             XML_3.99-0.3
## [41] zlibbioc_1.32.0             scales_1.1.1
## [43] hms_0.5.3                   ProtGenerics_1.18.0
## [45] SummarizedExperiment_1.16.1 yaml_2.2.1
## [47] curl_4.3                    memoise_1.1.0
## [49] stringi_1.4.6               RSQLite_2.2.0
## [51] BiocParallel_1.20.1         rlang_0.4.7
## [53] pkgconfig_2.0.3             bitops_1.0-6
## [55] evaluate_0.14               lattice_0.20-41
## [57] labeling_0.3                GenomicAlignments_1.22.1
## [59] htmlwidgets_1.5.1.9001      bit_1.1-15.2
## [61] tidyselect_1.1.0            R6_2.4.1
## [63] generics_0.0.2              DelayedArray_0.12.3
## [65] DBI_1.1.0                   mgcv_1.8-31
## [67] pillar_1.4.6                haven_2.3.1
## [69] withr_2.2.0                 RCurl_1.98-1.2
## [71] modelr_0.1.8                crayon_1.3.4
## [73] BiocFileCache_1.10.2        plotly_4.9.2.9000
## [75] rmarkdown_2.3               progress_1.2.2
## [77] locfit_1.5-9.4              grid_3.6.3
## [79] readxl_1.3.1                data.table_1.12.8
## [81] blob_1.2.1                  reprex_0.3.0
## [83] digest_0.6.25               openssl_1.4.2
## [85] munsell_0.5.0               viridisLite_0.3.0
## [87] askpass_1.1
```