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

Astra S. Bryant, PhD

July 11, 2022

Contents

1 Introduction

The goal of this file is to pre-process the *Strongyloides stercoralis* RNA-seq datasets originally published by Stolzfus *et al* 2012 and Gonzalez Akimori *et al* 2021.

1.1 Update Notes

- RNA-seq expression data for *S. stercoralis* free-living males added on 11-03-21 (ASB).
- Re-ran Kallisto alignment with WBPS16 reference transcriptome on 11-06-21 (ASB).
- Force calls to BioMart to not use caching, and import Kallisto reads every time. This may disrupt preprocessing pipeline if the reference transcriptome version does not match the current version on ParaSite (11-11-21, ASB).
- Re-ran Kallisto alignment with WBPS17 reference transcriptome on 07-08-22 (ASB)

2 Pre-processing Methods Overview

Kallisto and custom R scripts were used to perform ultra-fast read mapping of raw reads to the *S. stercoalis* reference transcriptome (PRJEB528.WBPS17.mRNA_transcripts, downloaded from WormBase Parasite on 08 July 2022). 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 both members of the out-group (*S. papillosus* and *S. venezuelensis*). Annotation information is 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 digital 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 is saved. Non-discarded gene values are 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 is 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 Appendix I.

3.1 Kallisto read mapping

This shell script checks the qualitiy of the fastq files and performs an alignment to the *Strongyloides stercoralis* 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. For addditional instructions, see comments at head of script. Note: Since FLM RNA-seq data was added after initial release of the browser, Kallisto alignment was performed separately for these samples.

```
# This script checks the qualitiy of the fastq files and performs an alignment to the Strongyloides ste
# To run this 'shell script' you will need to open your terminal and navigate to the directory where th
# 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 readMapp
# Then type './readMapping_Ss_WBPS17.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. This step could potentially be skipped if i
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.WBPS17.mRNA_transcripts.index strongyloides_sterco

# 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

# Free-Living Females
```

```
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o ERR146941 -t 14 E
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o ERR146942 -t 14 E
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o ERR146943 -t 14 E

# L3i+ (Activated iL3s)
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o ERR146944 -t 14 E
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o ERR146945 -t 14 E
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o ERR146946 -t 14 E

# L3i
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o ERR146947 -t 14 E
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o ERR146948 -t 14 E
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o ERR146949 -t 14 E

# Post-Free-Living L1s
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o ERR146950 -t 14 E
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o ERR146951 -t 14 E
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o ERR146952 -t 14 E

# Post-Parasitic L1s
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o ERR146953 -t 14 E
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o ERR146954 -t 14 E
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o ERR146955 -t 14 E

# Post-Parasitic L1s
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o ERR146956 -t 14 E
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o ERR146957 -t 14 E
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o ERR146958 -t 14 E

# Parasitic Females
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o ERR146959 -t 14 E
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o ERR146960 -t 14 E
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o ERR146961 -t 14 E

# Free-living Males
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o SRR13343624 -t 14
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o SRR13343625 -t 14
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o SRR13343626 -t 14

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

echo "Finished"
```

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 generates and saves an object containing the transcripts per million data. In subsequent chunks, that file is loaded, and analysis progresses.

```
# load packages ----
suppressPackageStartupMessages({
  library(tidyverse)
  library(tximport) # BioConductor
```

```
  library(ensembldb) # BioConductor
  library(biomaRt) # BioConductor
  library(magrittr)
})
# read in the study design ----
targets <- read_tsv("../Data/S_stercoralis/Study_Design/SstercoralisRNAseq_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'),
               useCache = FALSE,
               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"))
```

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
- *S. venezuelensis* homologs/percent homology
- UniProtKB number

- Interpro terms
- GO terms
- general Description information using biomart.

```
# 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,
# S. venezuelensis 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)
})

# 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'),
                     useCache = FALSE,
                     value = list('ststerprjeb528')) %>%
  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 First 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'),
                     useCache = FALSE,
                     value = list('ststerprjeb528')) %>%
  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 Second Out-subclade group homologs for S. stercoralis
# genes from BioMart and filter -----
Annt.temp.3 <- getBM(attributes=c('wbps_gene_id',
                                  'stveneprjeb530_gene',
                                  'stveneprjeb530_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'),
                     useCache = FALSE,
                     value = list('ststerprjeb528')) %>%
  as_tibble() %>%
  #rename columns
  dplyr::rename(geneID = wbps_gene_id,
                Out2.subclade_geneID = stveneprjeb530_gene,
                Out2.subclade_percent_homology= stveneprjeb530_homolog_perc_id
  ) %>%
  dplyr::group_by(geneID)

# Get C. elegans homologs and gene information for S. stercoralis
# genes from BioMart and filter -----
Annt.temp.4 <- 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'),
                     useCache = FALSE,
                     value = list('ststerprjeb528')) %>%
  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") %>%
  full_join(Annt.temp.4, 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.import$Out2.subclade_percent_homology[
  is.na(Annt.import$Out2.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) %>%
  slice_max(n = 1, order_by = Out2.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[
```

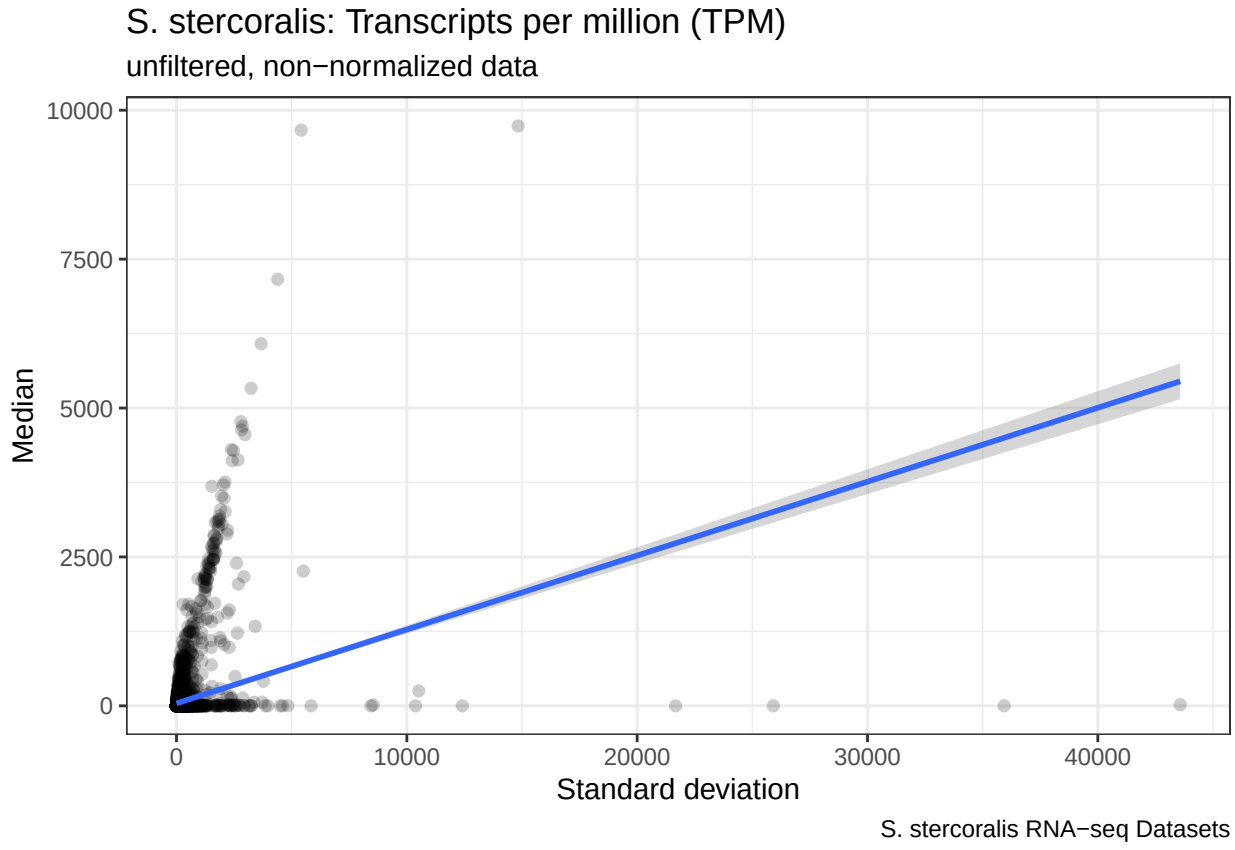
```
  Annt.logs$Ce_percent_homology == 1000] <- NA
Annt.logs$In.subclade_percent_homology[
  Annt.logs$In.subclade_percent_homology == 1000]<- NA
Annt.logs$Out.subclade_percent_homology[
  Annt.logs$Out.subclade_percent_homology == 1000]<- NA
Annt.logs$Out2.subclade_percent_homology[
  Annt.logs$Out2.subclade_percent_homology == 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,
                  Out2.subclade_geneID,
                  Out2.subclade_percent_homology,
                  .after = geneID) %>%
  column_to_rownames(var = "geneID")
```

3.4 Generate Digital Gene Expression List

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

```
# 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)
})
```

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

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

# 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 RNA-seq Datasets") +
  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)
```

3.5 Data Filtering and Normalization

The goal of this chunk is to:

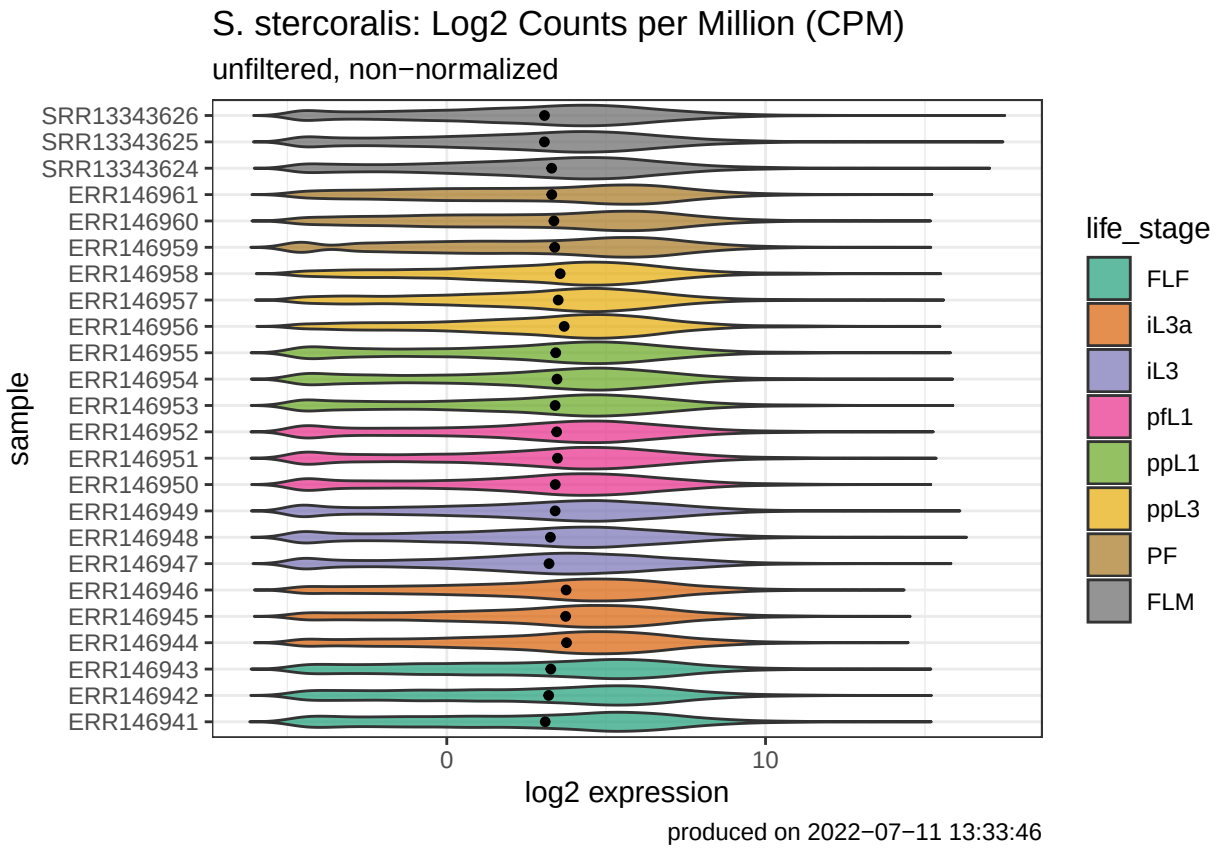
1. Filter and normalize data
2. Use `ggplot2` to visualize the impact of filtering and normalization on the data.

```
# 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
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()
```

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

```
p2
```

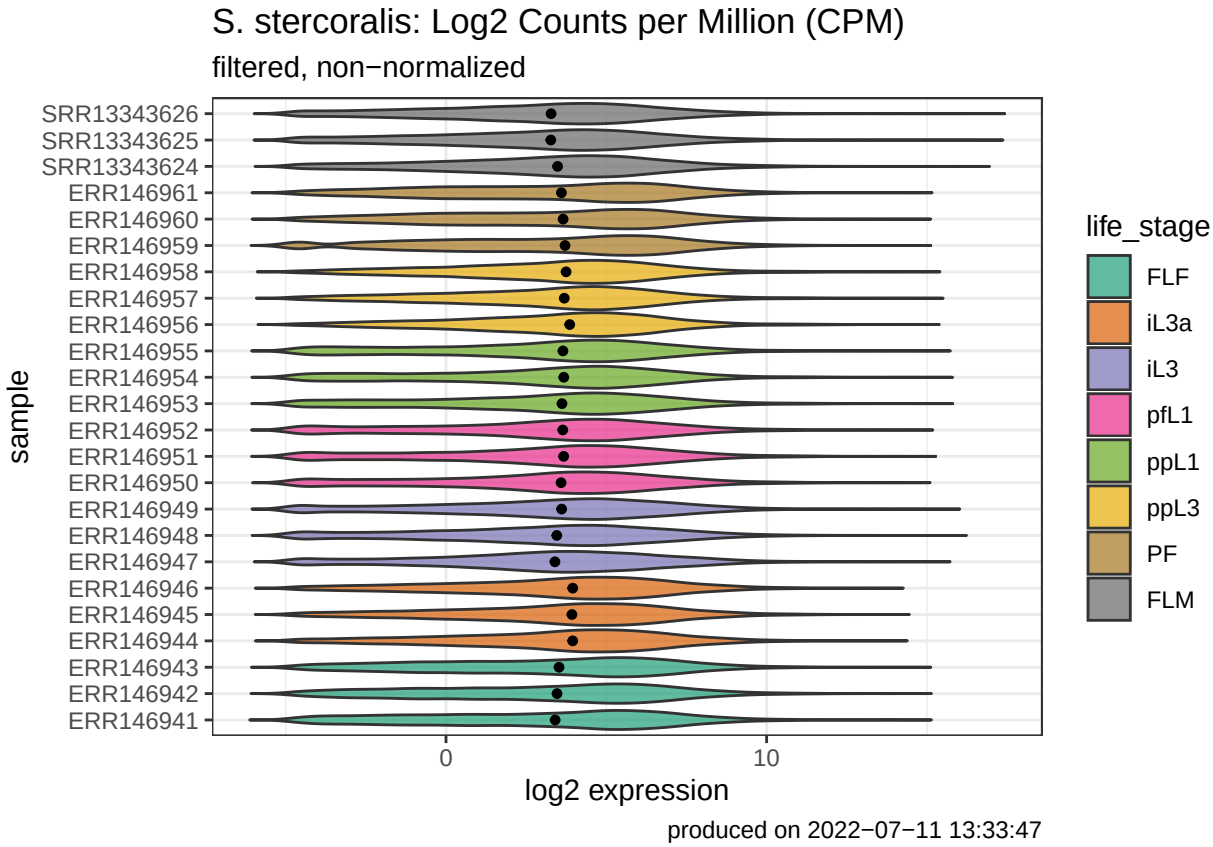



```
# Filter the data ----

# 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()
```

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

```
p3
```



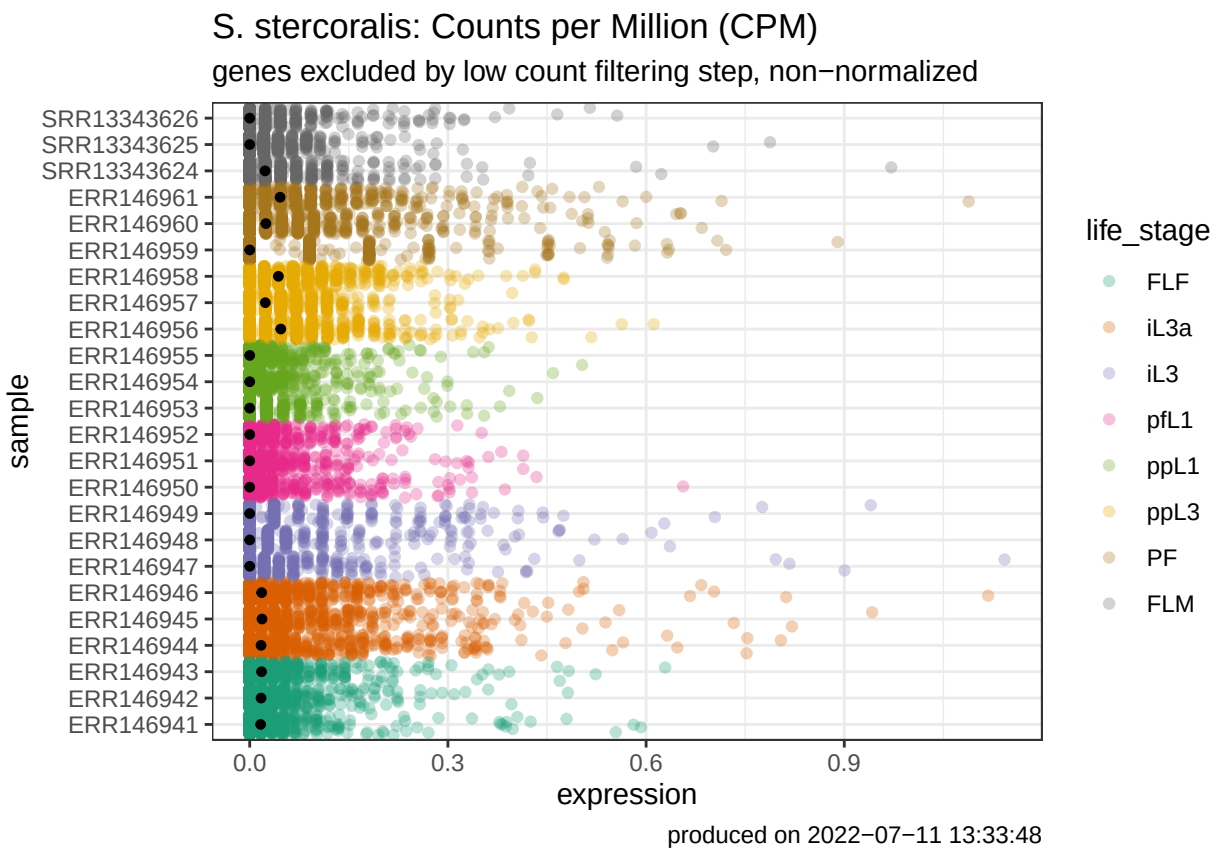

```
# Look at the genes excluded by the filtering step ----
# just to check that there aren't any with
# high expression that are in few samples
# Discarded genes
myDGEList.discarded <- myDGEList[!keepers,]

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()
```

3.5.3 Plot of genes discarded by low-copy filtering step

The low copy number filtering step excluded a total of 607 genes.

```
p.discarded
```



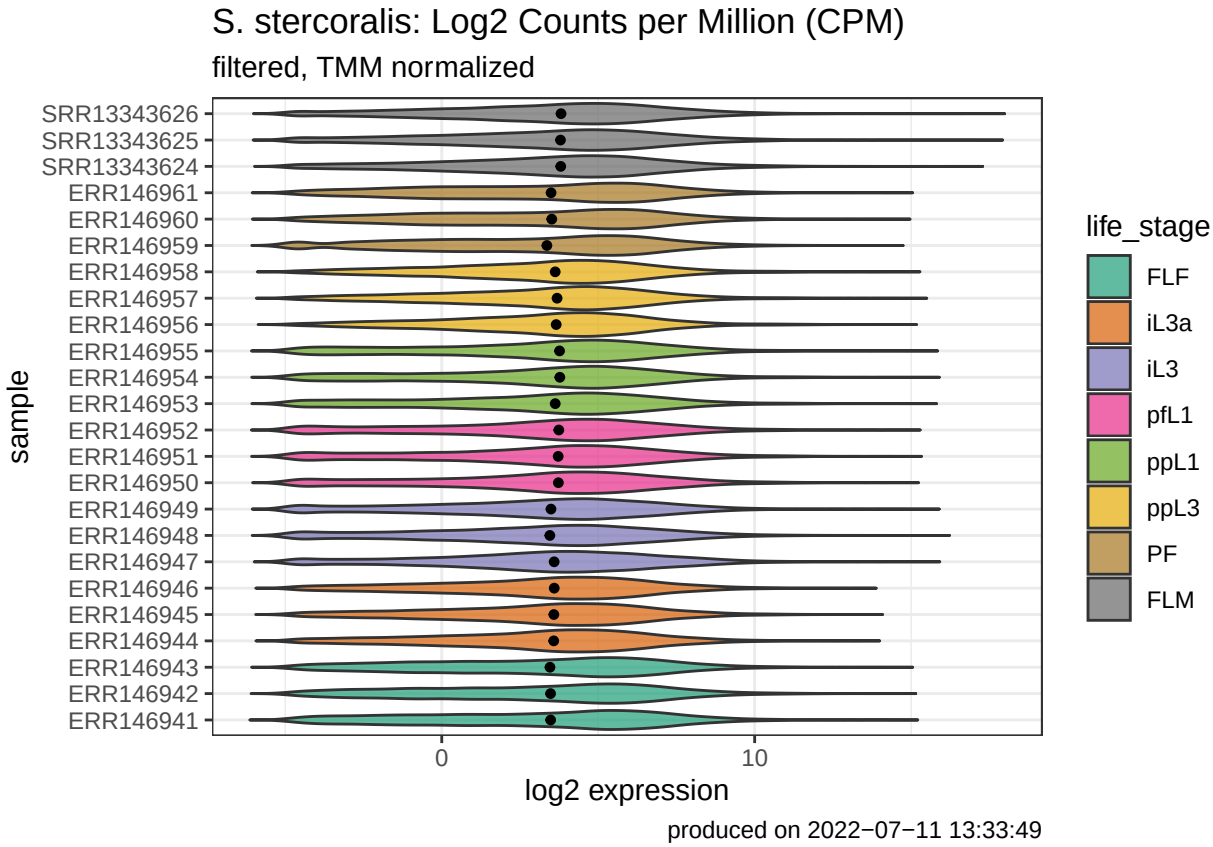

```
# # Carry out GO enrichment of discarded gene set using gProfiler2 ----
# discarded.geneID <- unique(log2.cpm.discarded.df.pivot$geneID)
# gost.res <- gost(list(Discarded_genes = discarded.geneID),
#                  organism = "stsisterprjeb528", correction_method = "fdr")
# gostplot(gost.res, interactive = 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()
```

3.5.4 Plot of filtered, normalized log2CPM data by life stage

```
p4
```

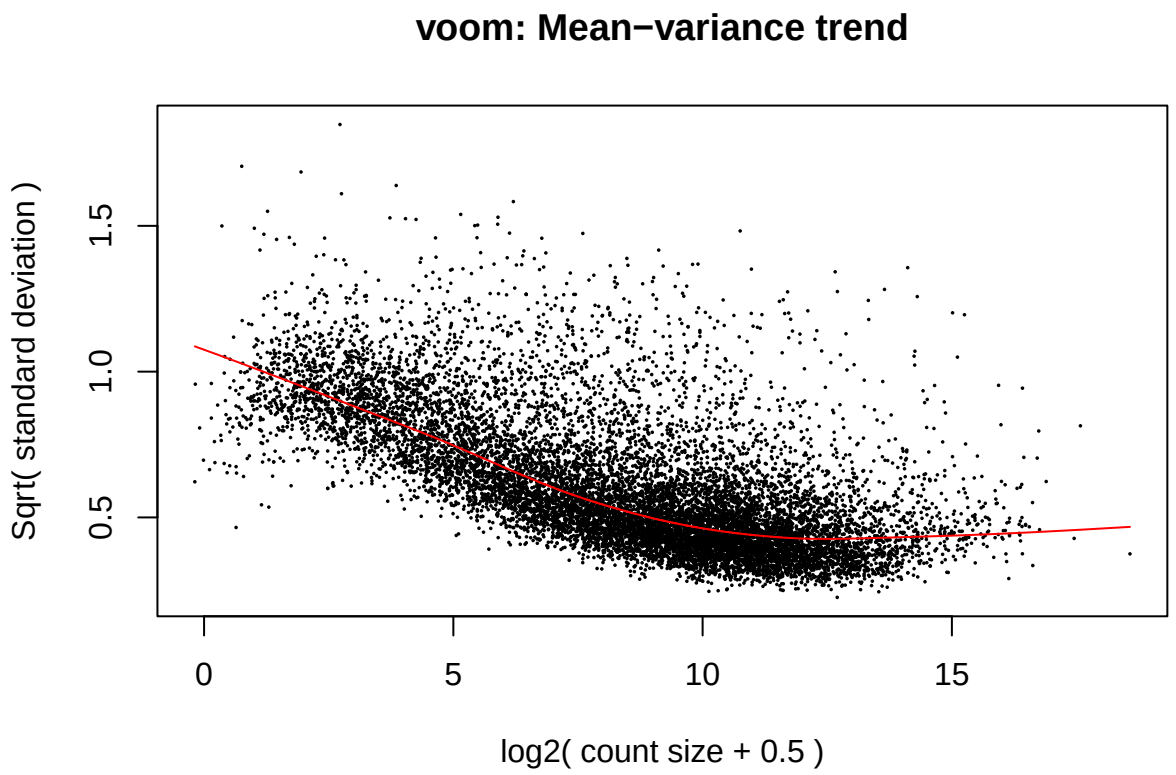


3.6 Compute Variance-Stabilized DGEList Object

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).

```
# 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 = '-')
```

3.7 Save Data and Annotations

Finally, we save data and annotations generated in code chunks above. We can separate these saving actions into two groups:

1. Data saved for downstream offline analyses, including the `SsRNAseq_data_preprocessed` file which saves filtered, normalized (but not voom adjusted) log2CPM values, gene annotation information, and sample information.
2. Files that are required inputs to the *Strongyloides* RNA-seq Browser App, including:
   i) a gene annotation R object (`Ss_geneAnnotations`)
   ii) the variance-stabilized vDGEList, saved as an R object (`Ss_vDGEList`)
   iii) a matrix of discarded genes and their raw counts (`SsRNAseq_discardedGene_counts.csv`) - this

data is downloadable from within the Browser App

iv) a matrix of variance-stabilized gene expression data, extracted from the vDGEList (`SsRNAseq_log2cpm_filtered_norm_voom.csv`) - this data is downloadable from within the Browser App

These files are saved in an Outputs folder; in order to make them accessible to a local version of the Shiny browser they need to be moved to appropriate subfolders within the App folder - the www sub folder (for .csv files) or the Data subfolder (for R objects). Stable copies are already located within those folders and do not need to be replaced unless the pre-processing steps change.

```
# 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 ----
# This object is required for the Shiny Browser
save(annotations,
file = file.path(output.path,
                         "Ss_geneAnnotations"))

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

# Save matrix of genes and their filtered, normalized, voom-transformed counts ----
# This is the count data that underlies the differential expression analyses in the Shiny app.
# Saving it here so that users of the app can access the input information.
write.csv(v.DEGList.filtered.norm$E,
          file = file.path(output.path,
                           "SsRNAseq_log2cpm_filtered_norm_voom.csv"))

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

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


# 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"))
```

4 Appendix I : All code for this report

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

# This script checks the qualitiy of the fastq files and performs an alignment to the Strongyloides ste
# To run this 'shell script' you will need to open your terminal and navigate to the directory where th
# 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 readMapp
# Then type './readMapping_Ss_WBPS17.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. This step could potentially be skipped if i
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.WBPS17.mRNA_transcripts.index strongyloides_sterco

# 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

# Free-Living Females
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o ERR146941 -t 14 E
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o ERR146942 -t 14 E
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o ERR146943 -t 14 E

# L3i+ (Activated iL3s)
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o ERR146944 -t 14 E
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o ERR146945 -t 14 E
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o ERR146946 -t 14 E

# L3i
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o ERR146947 -t 14 E
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o ERR146948 -t 14 E
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o ERR146949 -t 14 E

# Post-Free-Living L1s
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o ERR146950 -t 14 E
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o ERR146951 -t 14 E
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o ERR146952 -t 14 E

# Post-Parasitic L1s
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o ERR146953 -t 14 E
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o ERR146954 -t 14 E
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o ERR146955 -t 14 E

# Post-Parasitic L1s
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o ERR146956 -t 14 E
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o ERR146957 -t 14 E
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o ERR146958 -t 14 E
```

```
# Parasitic Females
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o ERR146959 -t 14 E
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o ERR146960 -t 14 E
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o ERR146961 -t 14 E

# Free-living Males
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o SRR13343624 -t 14
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o SRR13343625 -t 14
kallisto quant -i strongyloides_stercoralis.PRJEB528.WBPS17.mRNA_transcripts.index -o SRR13343626 -t 14

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

echo "Finished"

# load packages ----
suppressPackageStartupMessages({
  library(tidyverse)
  library(tximport) # BioConductor
  library(ensembldb) # BioConductor
  library(biomaRt) # BioConductor
  library(magrittr)
})
# read in the study design ----
targets <- read_tsv("../Data/S_stercoralis/Study_Design/SstercoralisRNAseq_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'),
               useCache = FALSE,
               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,
# S. venezuelensis 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)
})

# 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'),
                     useCache = FALSE,
                     value = list('ststerprjeb528')) %>%
  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 First 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'),
                     useCache = FALSE,
                     value = list('ststerprjeb528')) %>%
  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 Second Out-subclade group homologs for S. stercoralis
# genes from BioMart and filter -----
Annt.temp.3 <- getBM(attributes=c('wbps_gene_id',
                                  'stveneprjeb530_gene',
                                  'stveneprjeb530_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'),
                     useCache = FALSE,
                     value = list('ststerprjeb528')) %>%
  as_tibble() %>%
  #rename columns
  dplyr::rename(geneID = wbps_gene_id,
                Out2.subclade_geneID = stveneprjeb530_gene,
                Out2.subclade_percent_homology= stveneprjeb530_homolog_perc_id
  ) %>%
  dplyr::group_by(geneID)

# Get C. elegans homologs and gene information for S. stercoralis
# genes from BioMart and filter -----
Annt.temp.4 <- 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'),
                     useCache = FALSE,
                     value = list('ststerprjeb528')) %>%
  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") %>%
  full_join(Annt.temp.4, 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.import$Out2.subclade_percent_homology[
  is.na(Annt.import$Out2.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) %>%
  slice_max(n = 1, order_by = Out2.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
Annt.logs$Out2.subclade_percent_homology[
  Annt.logs$Out2.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,
                  Out2.subclade_geneID,
                  Out2.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)
})

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

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

# 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 RNA-seq Datasets") +
  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
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 ----
# This object is required for the Shiny Browser
save(annotations,
file = file.path(output.path,
```

```
                        "Ss_geneAnnotations"))

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

# Save matrix of genes and their filtered, normalized, voom-transformed counts ----
# This is the count data that underlies the differential expression analyses in the Shiny app.
# Saving it here so that users of the app can access the input information.
write.csv(v.DEGList.filtered.norm$E,
          file = file.path(output.path,
                           "SsRNAseq_log2cpm_filtered_norm_voom.csv"))

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

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


# 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"))

sessionInfo()
```

5 Appendix II: Session Info

```
sessionInfo()
```

```
## R version 4.2.0 (2022-04-22)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Catalina 10.15.7
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.2/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.2/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    stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] gprofiler2_0.2.1        RColorBrewer_1.1-3      ggthemes_4.2.4
##  [4] cowplot_1.1.1           matrixStats_0.62.0      edgeR_3.38.1
##  [7] limma_3.52.2            magrittr_2.0.3          biomaRt_2.52.0
## [10] ensembldb_2.20.2        AnnotationFilter_1.20.0 GenomicFeatures_1.48.3
## [13] AnnotationDbi_1.58.0    Biobase_2.56.0          GenomicRanges_1.48.0
## [16] GenomeInfoDb_1.32.2     IRanges_2.30.0          S4Vectors_0.34.0
## [19] BiocGenerics_0.42.0     tximport_1.24.0         forcats_0.5.1
## [22] stringr_1.4.0           dplyr_1.0.9             purrr_0.3.4
## [25] readr_2.1.2             tidyr_1.2.0             tibble_3.1.7
## [28] ggplot2_3.3.6           tidyverse_1.3.1
##
## loaded via a namespace (and not attached):
##   [1] colorspace_2.1-0            rjson_0.2.21
##   [3] ellipsis_0.3.2              XVector_0.36.0
##   [5] fs_1.5.2                    rstudioapi_0.13
##   [7] farver_2.1.1                bit64_4.0.5
##   [9] fansi_1.0.3                 lubridate_1.8.0
##  [11] xml2_1.3.3                  splines_4.2.0
##  [13] codetools_0.2-18            cachem_1.0.6
##  [15] knitr_1.39                  jsonlite_1.8.0
##  [17] Rsamtools_2.12.0            broom_1.0.0
##  [19] dbplyr_2.2.1                png_0.1-7
##  [21] compiler_4.2.0              httr_1.4.3
##  [23] backports_1.4.1             assertthat_0.2.1
##  [25] Matrix_1.4-1                fastmap_1.1.0
##  [27] lazyeval_0.2.2              cli_3.3.0
##  [29] htmltools_0.5.2             prettyunits_1.1.1
##  [31] tools_4.2.0                 gtable_0.3.0
##  [33] glue_1.6.2                  GenomeInfoDbData_1.2.8
##  [35] rappdirs_0.3.3              Rcpp_1.0.8.3
##  [37] cellranger_1.1.0            vctrs_0.4.1
##  [39] Biostrings_2.64.0           nlme_3.1-158
##  [41] rtracklayer_1.56.1          xfun_0.31
##  [43] rvest_1.0.2                 lifecycle_1.0.1
##  [45] restfulr_0.0.15             XML_3.99-0.10
##  [47] zlibbioc_1.42.0             scales_1.2.0
##  [49] vroom_1.5.7                 hms_1.1.1
##  [51] MatrixGenerics_1.8.1        ProtGenerics_1.28.0
##  [53] parallel_4.2.0              SummarizedExperiment_1.26.1
##  [55] yaml_2.3.5                  curl_4.3.2
##  [57] memoise_2.0.1               stringi_1.7.6
##  [59] RSQLite_2.2.14              highr_0.9
##  [61] BiocIO_1.6.0                filelock_1.0.2
##  [63] BiocParallel_1.30.3         rlang_1.0.3
##  [65] pkgconfig_2.0.3             bitops_1.0-7
##  [67] evaluate_0.15               lattice_0.20-45
##  [69] labeling_0.4.2              htmlwidgets_1.5.4
##  [71] GenomicAlignments_1.32.0    bit_4.0.4
##  [73] tidyselect_1.1.2            R6_2.5.1
```

```
##  [75] generics_0.1.3             DelayedArray_0.22.0
##  [77] DBI_1.1.3                  mgcv_1.8-40
##  [79] pillar_1.7.0               haven_2.5.0
##  [81] withr_2.5.0                KEGGREST_1.36.2
##  [83] RCurl_1.98-1.7             modelr_0.1.8
##  [85] crayon_1.5.1               utf8_1.2.2
##  [87] plotly_4.10.0              BiocFileCache_2.4.0
##  [89] tzdb_0.3.0                 rmarkdown_2.14
##  [91] progress_1.2.2             locfit_1.5-9.5
##  [93] grid_4.2.0                 readxl_1.4.0
##  [95] data.table_1.14.2          blob_1.2.3
##  [97] reprex_2.0.1               digest_0.6.29
##  [99] munsell_0.5.0              viridisLite_0.4.0
```