1 piRNA and Degradome Count Generation

Bryan Teefy

12/20/2019

Load Required Libraries

```
library(dplyr)
library(reshape2)
library(ggplot2)
library(ggpubr)
```

Classifying Transcripts in the *Hydra* Transcriptome

To determine the category of transcript to which piRNAs and degradome reads align, transcripts were classified as TEs, ncRNAs, uncharacterized transcripts, and genes.

The *Hydra* transcriptome was BLASTed against the *Hydra* Repbase, Swissprot, and nr databases with an e-value of 1e-5.

HMMER suite 3.1b2 (February 2015, http://www.hmmer.org/) and Pfam v31.0 database were used to identify protein domains in the transcriptome using an e-value of 1e-6.

Uniprot protein descriptions were added to any transcripts that had a match in the Swissprot database using Uniprot's Retrieve ID/mapping tool (https://www.uniprot.org/uploadlists/).

Open reading frames were identified using Transdecoder.

Results are summarized in Table S1.

Load Transcriptome Annotation Matrix

```
Transcript_Characterization <- read.table("objects/Transcriptome_Annotation_Matrix.txt",
    sep = "\t", check.names = FALSE, header = TRUE)
```

Transposon Annotation

Transcripts that met the following criteria were classified as TEs:

Transcripts with significant similarities to entries in the Repbase database.

Transcripts with Swissprot protein descriptions or nr sequence descriptions containing the strings "transpos", "J/jerky", and "mobile element".

Transcripts with Pfam domain descriptions predicted to encode domains containing "transposase", "THAP", "DDE_Tnp", "_Tnp" or "*tnp*".

non-coding RNA (ncRNA) Annotation

We considered sequences non-coding RNAs if they were lacking TE annotation, Swissprot hit, nr hit, known PFAM domain, and an ORF equal to or greater than 100 amino acids. ORFs were predicted using Transdecoder using command TransDecoder.LongOrfs -S -t.

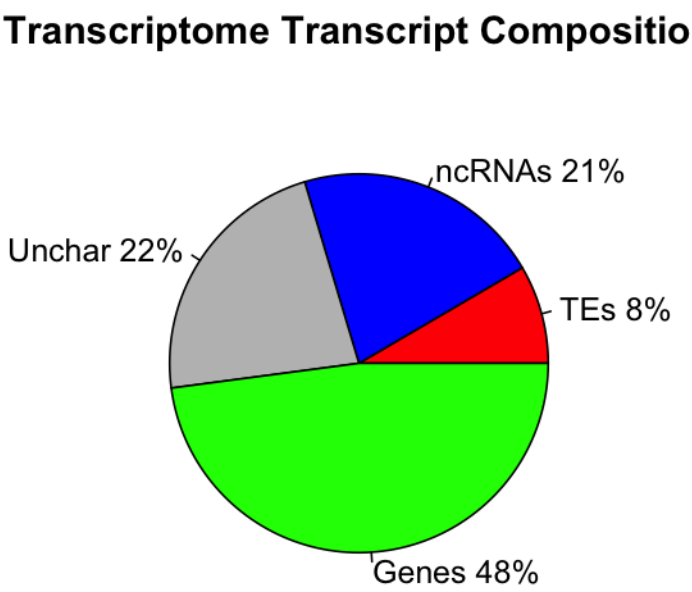
Taxonomically Restricted Genes (TRGs)/Uncharacterized Genes

Uncharacterized Genes were defined as transcripts predicted to contain an ORF equal to or greater than 100 amino acids without a Swissprot Hit, nr hit, known domain, or TE annotation. Nr hits termed "uncharacterized protein" were also considered in this category.

Gene Annotation

Genes were defined as transcripts with a Swissprot hit, nr hit, or domain annotation, and that were not classified as TEs by our annotation.

```
#Classify transcripts

Transcript_Characterization$Transcript_Class <- ifelse((

  !is.na(Transcript_Characterization$Repbase_Hit) | grepl(("transpos"), Transcript_Characterization$Uni
    grepl(("mobile element"), Transcript_Characterization$Uniprot_Description, ignore.case = TRUE) |
    grepl(("jerky"), Transcript_Characterization$Uniprot_Description, ignore.case = TRUE) |
    grepl(("transpos"), Transcript_Characterization$nr_Hit, ignore.case = TRUE) |
    grepl(("mobile element"), Transcript_Characterization$nr_Hit, ignore.case = TRUE) |
    grepl(("jerky"), Transcript_Characterization$nr_Hit, ignore.case = TRUE) |
    grepl(("Transposase"), Transcript_Characterization$PFAM_Annotation, ignore.case = TRUE) |
    grepl(("THAP"), Transcript_Characterization$PFAM_Annotation, ignore.case = TRUE) |
    grepl(("_Tnp_"), Transcript_Characterization$PFAM_Annotation, ignore.case = TRUE) |
    grepl(("DDE_Tnp"), Transcript_Characterization$PFAM_Annotation, ignore.case = TRUE) |
    grepl(("_Tnp"), Transcript_Characterization$PFAM_Annotation, ignore.case = TRUE)),"TE",

    ifelse(is.na(Transcript_Characterization$ORF) & is.na(Transcript_Characterization$Uniprot_Descriptio

    ifelse((!is.na(Transcript_Characterization$ORF) & is.na(Transcript_Characterization$Uniprot_Descrip

#Visualize Transcriptome Breakdown

transcriptome_annotation_whole <- c(sum(Transcript_Characterization$Transcript_Class == "TE"),sum(Transc

lbls <- c("TEs", "ncRNAs", "Unchar", "Genes")
colors = c("red", "blue", "gray", "green")
pct <- round(transcriptome_annotation_whole/sum(transcriptome_annotation_whole)*100)
lbls <- paste(lbls, pct) # add percents to labels
lbls <- paste(lbls,"%",sep="") # ad % to labels
pie(transcriptome_annotation_whole,labels = lbls, col=colors,
    main="Transcriptome Transcript Composition")
```



Generating piRNA and Degradome counts

Since piRNAs are complementary to RNA transcript targets (antisense to the target) or derived directly from targets (sense to the target), we used piRNA mapping to identify these targets in the *Hydra* transcriptome.

Adapters were trimmed from the WT and Colchicine raw reads using the script trimbiooadapter.sh.

piRNAs were mapped to the *Hydra* transcriptome with Bowtie v1.1.2 using the shell script: piRNA_Deg_Mapping.sh.
Three mismatches were allowed in the antisense orientation and no mismatches were allowed in the sense orientation. Degradome reads were mapped in the sense orientation with no mismatches since degradome reads are transcript fragments.

A count matrix consisting of the number of mapped piRNAs per transcript was generated using the script: Counting_Matrix_Gen.R.
Importantly this script apportioned multimapping piRNAs fractionally such that the count value for a particular piRNA mapping to a transcript was divided by the number of times the piRNA mapped to the transcriptome.

Counting_Matrix_Gen.R was run using the script: run_Counting_Matrix_Gen.sh.

Results are summarized in the table, “piRNA_Counts_Matrix.txt” and “Degradome_Counts_Matrix.txt”, which can be found in the GEO repository (GSE135440).

Load and Merge the piRNA and Degradome Count Files

```
piRNA_counts <- read.table("objects/piRNA_Count_Matrix.txt", header = T)

Deg_counts <- read.table("objects/Deg_Count_Matrix.txt", header = T)

piRNA_Deg_counts <- merge(piRNA_counts, Deg_counts, by = "ID")

piRNA_Deg_counts <- merge(Transcript_Characterization, piRNA_Deg_counts, by = "ID")
```

Generate Normalized piRNA Counts

PIWI targets should have a high density of piRNA counts. We normalize piRNA counts by transcript length to determine piRNA count density (Reads per kilobase [RPK]).

To determine if piRNA count density values were significantly different between classes of transcripts, we performed Tukey's Honest Significant Difference test to compare mean piRNA count density between each transcript type (i.e. TE, ncRNA, Unchar., Gene) for each piRNA class (i.e. Hywi Antisense-mapped, Hyli Sense-mapped).

```
# Generate RPK values

norm <- (piRNA_Deg_counts$Length/1000)

piRNA_Deg_counts[, c(22:31)] <- piRNA_Deg_counts[, c(12:21)]/norm

colnames(piRNA_Deg_counts)[22:31] <- c("WT_Hywi_AS_Counts_RPK", "WT_Hyli_AS_Counts_RPK",
    "WT_Hywi_S_Counts_RPK", "WT_Hyli_S_Counts_RPK", "Colch_Hywi_AS_Counts_RPK", "Colch_Hyli_AS_Counts_RP
    "Colch_Hywi_S_Counts_RPK", "Colch_Hyli_S_Counts_RPK", "WT_Deg_Counts_RPK", "Colch_Deg_Counts_RPK")


# Perform Tukey's Honest Significant Difference test

# Group normalized mapping counts
Normalized_Mapping_Counts_Matrix <- piRNA_Deg_counts[, c(22:29, 11)]
Normalized_Mapping_Counts_Matrix_Formatted <- melt(Normalized_Mapping_Counts_Matrix,
    id.var = "Transcript_Class")

# Subset count density based on piRNA origin
WT_Hyli_AS_Counts_Stats <- subset(Normalized_Mapping_Counts_Matrix_Formatted, variable ==
    "WT_Hyli_AS_Counts_RPK")
WT_Hyli_S_Counts_Stats <- subset(Normalized_Mapping_Counts_Matrix_Formatted, variable ==
    "WT_Hyli_S_Counts_RPK")
WT_Hywi_AS_Counts_Stats <- subset(Normalized_Mapping_Counts_Matrix_Formatted, variable ==
    "WT_Hywi_AS_Counts_RPK")
WT_Hywi_S_Counts_Stats <- subset(Normalized_Mapping_Counts_Matrix_Formatted, variable ==
    "WT_Hywi_S_Counts_RPK")

Colch_Hyli_AS_Counts_Stats <- subset(Normalized_Mapping_Counts_Matrix_Formatted,
    variable == "Colch_Hyli_AS_Counts_RPK")
Colch_Hyli_S_Counts_Stats <- subset(Normalized_Mapping_Counts_Matrix_Formatted, variable ==
    "Colch_Hyli_S_Counts_RPK")
```

```r
Colch_Hywi_AS_Counts_Stats <- subset(Normalized_Mapping_Counts_Matrix_Formatted,
    variable == "Colch_Hywi_AS_Counts_RPK")
Colch_Hywi_S_Counts_Stats <- subset(Normalized_Mapping_Counts_Matrix_Formatted, variable ==
    "Colch_Hywi_S_Counts_RPK")

# Develop Tukey Test Function (ANOVA post hoc test)

Tukey_Test <- function(x) {
    res.aov <- aov(value ~ Transcript_Class, data = x)
    return(TukeyHSD(res.aov))
}

# Run Tukey Test

Tukey_Test(WT_Hyli_AS_Counts_Stats)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = value ~ Transcript_Class, data = x)
##
## $Transcript_Class
##                     diff         lwr         upr     p adj
## ncRNA-Gene     676.8018   411.99715   941.60647 0.0000000
## TE-Gene       2367.7919  1988.36297  2747.22084 0.0000000
## Unchar-Gene    320.8933    61.26345   580.52305 0.0081561
## TE-ncRNA      1690.9901  1277.40463  2104.57557 0.0000000
## Unchar-ncRNA  -355.9086  -663.30774   -48.50937 0.0155491
## Unchar-TE    -2046.8987 -2457.19010 -1636.60722 0.0000000
```

```r
Tukey_Test(WT_Hyli_S_Counts_Stats)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = value ~ Transcript_Class, data = x)
##
## $Transcript_Class
##                     diff        lwr        upr     p adj
## ncRNA-Gene     641.7496   326.2192   957.2800 0.0000010
## TE-Gene       3136.1628  2684.0509  3588.2748 0.0000000
## Unchar-Gene   1133.4406   824.0764  1442.8048 0.0000000
## TE-ncRNA      2494.4132  2001.6017  2987.2247 0.0000000
## Unchar-ncRNA   491.6910   125.4067   857.9753 0.0031614
## Unchar-TE    -2002.7222 -2491.6087 -1513.8358 0.0000000
```

```r
Tukey_Test(WT_Hywi_AS_Counts_Stats)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = value ~ Transcript_Class, data = x)
##
## $Transcript_Class
##                     diff          lwr        upr     p adj
```

```
## ncRNA-Gene     550.6539   412.623440   688.6844 0.0000000
## TE-Gene       2456.9401  2259.161224  2654.7189 0.0000000
## Unchar-Gene    719.4290   584.095902   854.7620 0.0000000
## TE-ncRNA      1906.2861  1690.703074  2121.8692 0.0000000
## Unchar-ncRNA   168.7750     8.542001   329.0081 0.0343943
## Unchar-TE    -1737.5111 -1951.377135 -1523.6451 0.0000000
```

```
Tukey_Test(WT_Hywi_S_Counts_Stats)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = value ~ Transcript_Class, data = x)
##
## $Transcript_Class
##                     diff        lwr        upr     p adj
## ncRNA-Gene     15.47210  -22.13980   53.08400 0.7158328
## TE-Gene       357.69360  303.80088  411.58633 0.0000000
## Unchar-Gene    31.34943   -5.52745   68.22632 0.1276507
## TE-ncRNA      342.22150  283.47731  400.96570 0.0000000
## Unchar-ncRNA   15.87733  -27.78454   59.53921 0.7864508
## Unchar-TE    -326.34417 -384.62049 -268.06785 0.0000000
```

```
Tukey_Test(Colch_Hyli_AS_Counts_Stats)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = value ~ Transcript_Class, data = x)
##
## $Transcript_Class
##                     diff        lwr         upr     p adj
## ncRNA-Gene     380.7634   256.5542  504.972613 0.0000000
## TE-Gene       1406.5476  1228.5728 1584.522493 0.0000000
## Unchar-Gene    226.6174   104.8355  348.399343 0.0000104
## TE-ncRNA      1025.7842   831.7879 1219.780568 0.0000000
## Unchar-ncRNA  -154.1460  -298.3346   -9.957341 0.0306720
## Unchar-TE    -1179.9302 -1372.3814 -987.478957 0.0000000
```

```
Tukey_Test(Colch_Hyli_S_Counts_Stats)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = value ~ Transcript_Class, data = x)
##
## $Transcript_Class
##                     diff        lwr       upr     p adj
## ncRNA-Gene     17.99663  -72.48296  108.4762 0.9565031
## TE-Gene       767.67172  638.02683  897.3166 0.0000000
## Unchar-Gene   189.64385  100.93243  278.3553 0.0000002
## TE-ncRNA      749.67509  608.35945  890.9907 0.0000000
## Unchar-ncRNA  171.64722   66.61376  276.6807 0.0001578
## Unchar-TE    -578.02788 -718.21800 -437.8377 0.0000000
```

```
Tukey_Test(Colch_Hywi_AS_Counts_Stats)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = value ~ Transcript_Class, data = x)
##
## $Transcript_Class
##                      diff        lwr        upr     p adj
## ncRNA-Gene     522.48515   268.8335   776.1368 0.0000007
## TE-Gene       3389.43088  3025.9827  3752.8791 0.0000000
## Unchar-Gene    492.18109   243.4863   740.8758 0.0000022
## TE-ncRNA      2866.94573  2470.7796  3263.1119 0.0000000
## Unchar-ncRNA   -30.30407  -324.7563   264.1481 0.9935328
## Unchar-TE    -2897.24979 -3290.2606 -2504.2390 0.0000000
```

```
Tukey_Test(Colch_Hywi_S_Counts_Stats)
```

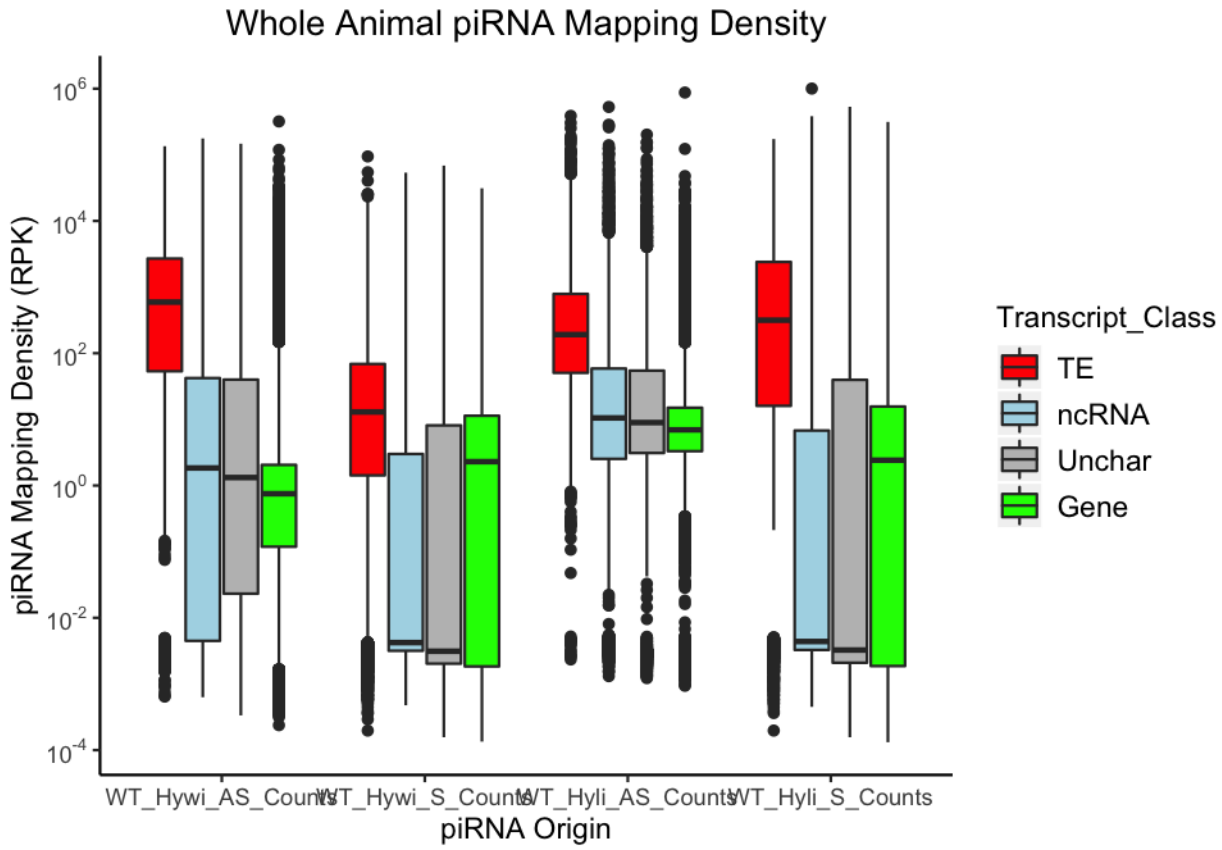
```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = value ~ Transcript_Class, data = x)
##
## $Transcript_Class
##                      diff        lwr         upr     p adj
## ncRNA-Gene    -70.675926 -129.30100  -12.050851 0.0105377
## TE-Gene       171.966758   87.96503  255.968488 0.0000009
## Unchar-Gene   -67.337842 -124.81725   -9.858429 0.0139294
## TE-ncRNA      242.642684  151.07904  334.206327 0.0000000
## Unchar-ncRNA    3.338085  -64.71699   71.393157 0.9992856
## Unchar-TE    -239.304599 -330.13898 -148.470222 0.0000000
```

Visualizing piRNA Mapping

Since the range of observed count density values was large, we used a log scale to visualize piRNA count density. For boxplot visualization, we added a pseudocount to the raw piRNA counts to remove any 0 count density values that would return infinite values on a log scale. The pseudocount we chose was 0.001 since that approximated the lowest fractional count used in our counting strategy. We explored piRNA count density for 1) Whole animals and 2) Epithelial Animals.

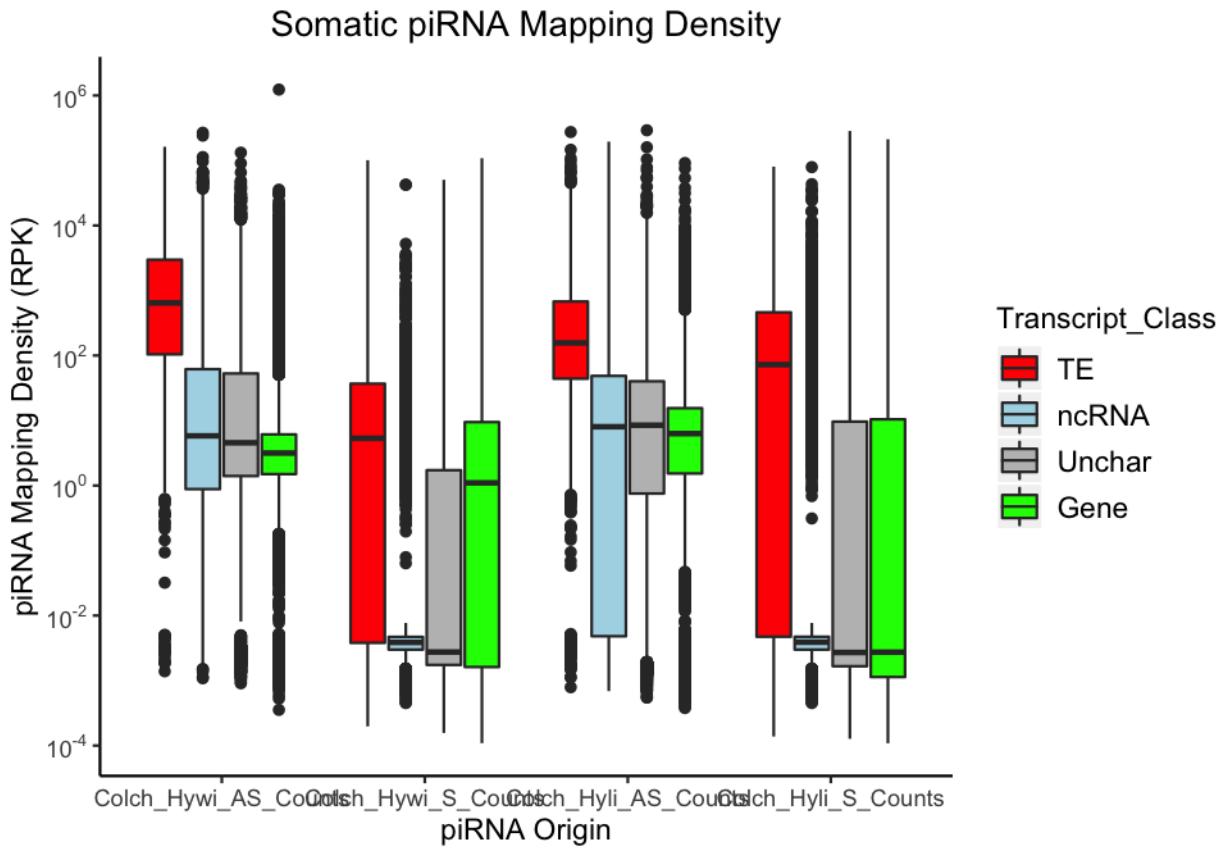
```
# Set pseudocount

pseudocount <- 0.001

# Add pseudocount to raw piRNA counts then generate piRNA count density values

boxplot_matrix <- piRNA_Deg_counts[, c(12:19, 3, 11)]
boxplot_matrix[, c(1:8)] <- boxplot_matrix[, c(1:8)] + pseudocount
boxplot_matrix[, c(1:8)] <- boxplot_matrix[, c(1:8)]/(boxplot_matrix$Length/1000)

# Plot whole animal piRNA count density values

wt_matrix_data <- boxplot_matrix[, c(10, 1:4)]
wt_matrix_data_formatted <- melt(wt_matrix_data, id.var = "Transcript_Class")
colnames(wt_matrix_data_formatted) <- c("Transcript_Class", "piRNA_Origin", "piRNA_Mapping_Density")
wt_matrix_data_formatted$Transcript_Class <- factor(wt_matrix_data_formatted$Transcript_Class,
```

```
    levels = c("TE", "ncRNA", "Unchar", "Gene"))

WT_level_order <- c("WT_Hywi_AS_Counts", "WT_Hywi_S_Counts", "WT_Hyli_AS_Counts",
    "WT_Hyli_S_Counts")

WT_boxplot <- ggplot(data = wt_matrix_data_formatted, aes(x = factor(piRNA_Origin,
    level = WT_level_order), y = piRNA_Mapping_Density), log = "y") + geom_boxplot(aes(fill = Transcript
    scale_y_log10(breaks = scales::trans_breaks("log10", function(x) 10^x), labels = scales::trans_forma
        scales::math_format(10^.x)))

WT_boxplot + scale_fill_manual(values = c("red", "light blue", "grey", "green")) +
    theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(), axis.line = element_line(colour = "black")) +
    theme(legend.text = element_text(size = rel(1))) + ggtitle("Whole Animal piRNA Mapping Density") +
    theme(plot.title = element_text(hjust = 0.5)) + xlab("piRNA Origin") + ylab("piRNA Mapping Density
```




```
# Plot epithelial animal piRNA count density values

colch_matrix_data <- boxplot_matrix[, c(10, 5:8)]
colch_matrix_data_formatted <- melt(colch_matrix_data, id.var = "Transcript_Class")
colnames(colch_matrix_data_formatted) <- c("Transcript_Class", "piRNA_Origin", "piRNA_Mapping_Density")
colch_matrix_data_formatted$Transcript_Class <- factor(colch_matrix_data_formatted$Transcript_Class,
    levels = c("TE", "ncRNA", "Unchar", "Gene"))

colch_level_order <- c("Colch_Hywi_AS_Counts", "Colch_Hywi_S_Counts", "Colch_Hyli_AS_Counts",
    "Colch_Hyli_S_Counts")
```

```
colch_boxplot <- ggplot(data = colch_matrix_data_formatted, aes(x = factor(piRNA_Origin,
    level = colch_level_order), y = piRNA_Mapping_Density), log = "y") + geom_boxplot(aes(fill = Transc
    scale_y_log10(breaks = scales::trans_breaks("log10", function(x) 10^x), labels = scales::trans_forma
        scales::math_format(10^.x)))

colch_boxplot + scale_fill_manual(values = c("red", "light blue", "grey", "green")) +
    theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(), axis.line = element_line(colour = "black")) +
    theme(legend.text = element_text(size = rel(1))) + ggtitle("Somatic piRNA Mapping Density") +
    theme(plot.title = element_text(hjust = 0.5)) + xlab("piRNA Origin") + ylab("piRNA Mapping Density
```




```
# write table that summarizes results write.table(piRNA_Deg_counts, file =
# 'objects/Annotated_piRNA_Degradome_Count_Matrix.txt', sep = '\t')
```

Assessing piRNA Origin

piRNAs can be derived from distinct genomic loci, such as dedicated piRNA clusters, or from antisense transcription. If piRNAs are dervied from antisense transcription, piRNAs should map to such targets in the *Hydra* transcriptome in the antisense orientation without mismatches. However, if piRNAs are generated from distinct genomic loci, it is likely that piRNAs will align to their targets with mismatches.

To address this distinction, we mapped piRNAs in the antisense orientation without mismatches to the *Hydra* transcriptome using the script: No_mismatch_mapping.sh

The script "piRNA_stats.sh" was used to calculate the total number of aligned piRNAs with and without

mismatches. The results showed that 68.0% of piRNAs isolated from WT animals and 85.1% of piRNAs isolated from epithelial animals mapped with at least one mismatch. This indicates that the majority of piRNAs are dervied from a locus distinct from the target transcript.

Software versions

This document was computed on Thu Jan 09 15:18:14 2020 with the following R package versions.

```
R version 3.5.3 (2019-03-11)
Platform: x86_64-apple-darwin15.6.0 (64-bit)
Running under: macOS Mojave 10.14.5

Matrix products: default
BLAS: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/3.5/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] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] ggpubr_0.2.4   magrittr_1.5   ggplot2_3.2.0  reshape2_1.4.3
[5] dplyr_0.8.3    knitr_1.22

loaded via a namespace (and not attached):
 [1] Rcpp_1.0.1       munsell_0.5.0    tidyselect_0.2.5 colorspace_1.4-1
 [5] R6_2.4.0         rlang_0.4.0      stringr_1.4.0    plyr_1.8.4
 [9] tools_3.5.3      grid_3.5.3       gtable_0.3.0     xfun_0.5
[13] withr_2.1.2      htmltools_0.3.6  lazyeval_0.2.2   yaml_2.2.0
[17] assertthat_0.2.1 digest_0.6.20    tibble_2.1.3     ggsignif_0.5.0
[21] crayon_1.3.4     purrr_0.3.2      formatR_1.7      glue_1.3.1
[25] evaluate_0.13    rmarkdown_1.12   stringi_1.4.3    compiler_3.5.3
[29] pillar_1.4.2     scales_1.0.0     pkgconfig_2.0.2
```

2 Differential Gene Expression Analysis and GO Enrichment

Stefan Siebert and Bryan Teefy

12/20/2019

Differential Gene Expression

We performed differential gene expression analysis between wildtype and *hywi* knockdown animals. As a first step we generated expression estimates using RSEM/bowtie and used reads that were filtered for adapters (TruSeq3) using trimmomatic (LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15 MINLEN:36). We made use of the RSEM functions rsem-calculate-expression (–forward-prob 0) and rsem-generate-data-matrix to generate an expression matrix (scripts: DGE_expression.sh, DGE_count_matrix.sh). The raw count matrix is available at GEO (GSE135440).

Load the expression data / GO annotations

```
# Load the Differential Gene Expression Count Matrix

counts <- read.table("objects/Differential_Gene_Expression_Count_Matrix.txt", sep = "\t",
    check.names = FALSE, header = TRUE, row.names = 1)

# Load normalized piRNA and degradome reads and transcript characterization from
# RMD1

piRNA_Deg_counts <- read.table("objects/Annotated_piRNA_Degradome_Count_Matrix.txt",
    sep = "\t")
```

Load Required Packages

```
library(edgeR)
library(knitr)
library(xtable)
library(ggplot2)
```

Data exploration

We explore the data to get insights into the paired nature of the triplicate samples from wildtype and *hywi* knockdown tissue.

```
# We first calculate normalized counts for future use and to include them in the
# master dataframe.

# set gene IDs as rownames
rownames(counts) <- counts[, 1]
```

```r
# raw counts from all treatments
k <- counts[, c(2:7)]

# calculate normalization factors
nf_k <- calcNormFactors(k)

# calculate library sizes
ls_k <- colSums(k)

# effective library size using normalization factors
lse_k <- ls_k * nf_k

# normalization multiplier to use on counts
nm_k <- 1e+06/lse_k

# normalize counts using normalization multiplier
k <- k * nm_k

# round normalized counts
k <- round(k, digits = 0)

# restore ID column
k$ID <- rownames(k)

# reorder columns
k <- k[, c(7, 1:6)]

# combine rounded raw and normalized counts
k <- merge(counts[, c(1:7)], k, by = "ID")

# rename columns

colnames(k) <- c("ID", "WT1", "WT2", "WT3", "KD1", "KD2", "KD3", "nWT1", "nWT2",
    "nWT3", "nKD1", "nKD2", "nKD3")
```

```r
# explore replication

# define treatment groups for DGE
TR_k <- factor(c("k", "k", "k", "w", "w", "w"))

# set wild type as reference
TR_k <- relevel(TR_k, ref = "w")

# create experimental design data frame defining treatment type for each sample
d_k <- data.frame(Sample = colnames(counts[, c(5, 6, 7, 2, 3, 4)]), TR_k)

# generate DGEList object containing raw counts, the treatment type for each
# column, and the gene IDs
y <- DGEList(counts = counts[, c(5, 6, 7, 2, 3, 4)], group = d_k$TR_k, genes = counts[,
    1])

# label each column with the appropriate sample name
colnames(y) <- d_k$Sample
```

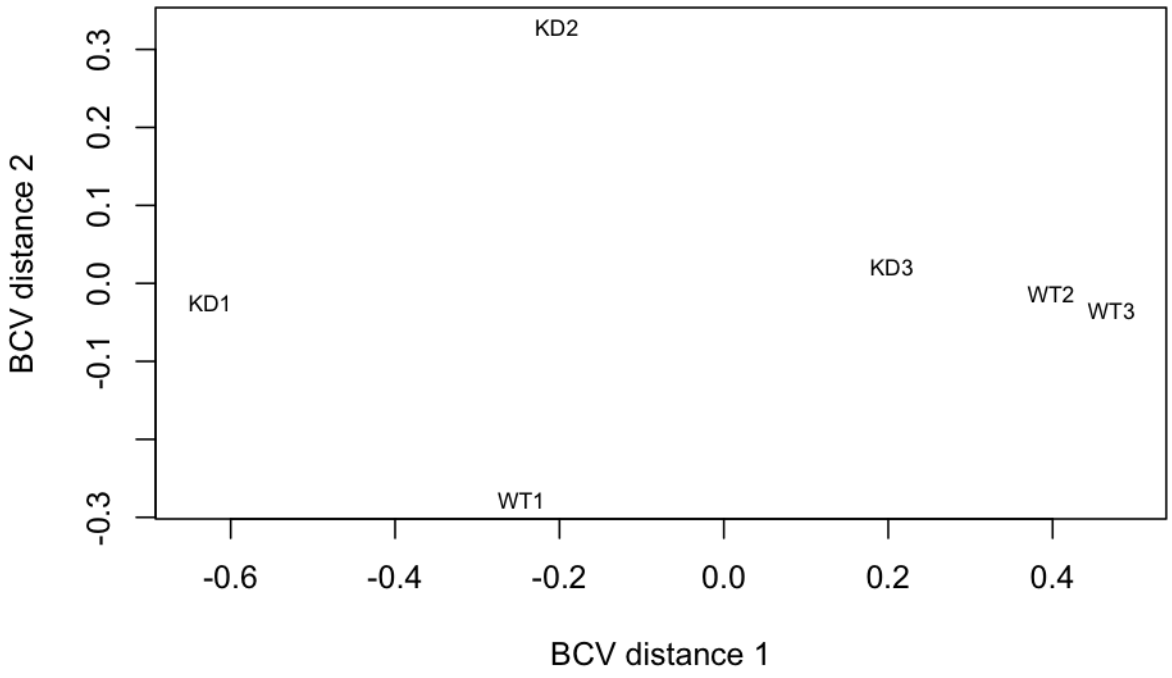
```r
# calculate library size for each sample and store within DGElist
y$samples$lib.size <- colSums(y$counts)

# exclude transcripts that do not have at least two samples with more than one
# count per million
keep <- rowSums(cpm(y) > 1) >= 2
y <- y[keep, , keep.lib.sizes = FALSE]

# number of transcripts remaining after count filtering
dim(y)
```

```
## [1] 16435     6
```

```r
# calculate normalization factors for each sample
y <- calcNormFactors(y)

# review library sizes and normalization factors
y$samples
```

```
##     group lib.size norm.factors
## KD1     k 27236412    1.0711124
## KD2     k 26955915    0.9927897
## KD3     k 27081381    1.0126300
## WT1     w 22165507    1.0181741
## WT2     w 27471090    0.9836625
## WT3     w 25597005    0.9272326
```

```r
# create a matrix defining the control and treatment groups for the analysis
# based on what is described by the TR object
design_k <- model.matrix(~TR_k)

# Estimate dispersions
y <- estimateDisp(y, design_k)

# generate MDS Plot

plotMDS(y, method = "bcv", cex = 0.7)
```




```
# we exclude outlier replicates WT1 and KD3 from downstream analyses

# define treatment groups for DGE
TR_k <- factor(c("k", "k", "w", "w"))

# set wild type as reference
TR_k <- relevel(TR_k, ref = "w")

# create experimental design data frame defining treatment type for each sample
d_k <- data.frame(Sample = colnames(counts[, c(5, 6, 3, 4)]), TR_k)

# generate DGElist object containing raw counts, the treatment type for each
# column, and the gene annotations
y <- DGEList(counts = counts[, c(5, 6, 3, 4)], group = d_k$TR_k, genes = counts[,
    1])

# label each column with the appropriate sample name
colnames(y) <- d_k$Sample

# calculate library size for each sample and store within DGElist
y$samples$lib.size <- colSums(y$counts)

# exclude transcripts that do not have at least two samples with more than one
# count per million
keep <- rowSums(cpm(y) > 1) >= 2
y <- y[keep, , keep.lib.sizes = FALSE]
```

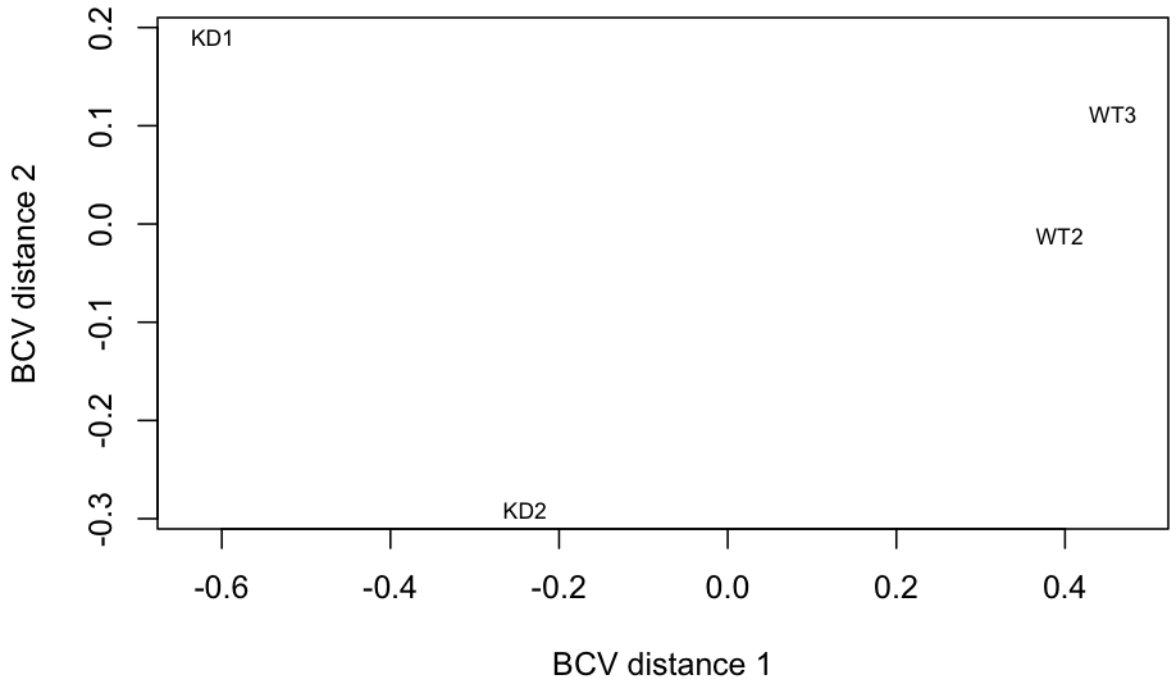
```r
# number of transcripts remaining after count filtering
dim(y)
```

```
## [1] 15924     4
```

```r
# calculate normalization factors for each sample
y <- calcNormFactors(y)

# review library sizes and normalization factors
y$samples
```

```
##     group lib.size norm.factors
## KD1     k 27217655    1.0849445
## KD2     k 26945252    0.9960237
## WT2     w 27460768    0.9922666
## WT3     w 25588479    0.9325978
```

```r
# create a matrix defining the control and treatment groups for the analysis
# based on what is described by the TR object
design_k <- model.matrix(~TR_k)

# estimate dispersions
y <- estimateDisp(y, design_k)

# generate MDS Plot
plotMDS(y, method = "bcv", cex = 0.7)
```



Differential Gene Expression (DGE) analysis - wildtype vs *hywi* knockdown

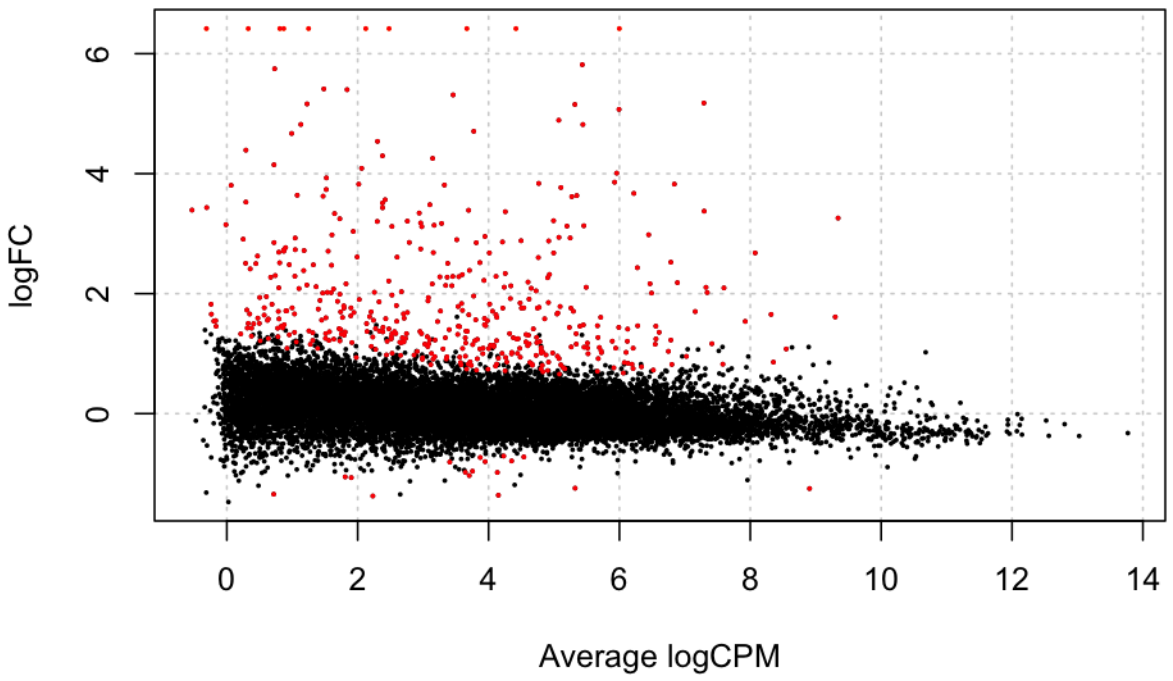
We perform DGE analysis after excluding outlier replicates WT1 and KD3.

```
# DGE Analysis

# set p-value for cutoff
p.value = 0.05

# fit data to a negative binomial geralized log-linear model
fit_k <- glmFit(y, design_k)

# conduct a statistical test for differential gene expression based on the fit
# from
lrt_k <- glmLRT(fit_k)

# create a list of values that describes the status of each gene in the DGE test
de_k <- decideTestsDGE(lrt_k, adjust.method = "BH", p.value)

# overview of number of differentially expressed genes
summary(de_k)
```

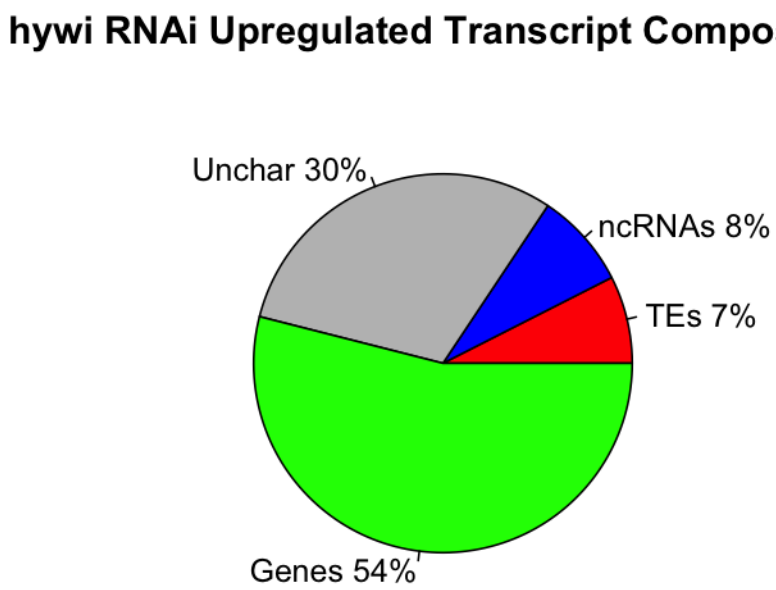
```
##        TR_kk
## Down      17
## NotSig 15466
## Up       441
```

```
# build results table
D_k <- lrt_k$table

# restore ID column
D_k$ID <- rownames(D_k)

# add column of adjusted p values
D_k <- cbind(D_k, p.adjust(D_k$PValue, method = "BH"))
names(D_k)[names(D_k) == "p.adjust(D_k$PValue, method = \"BH\")"] = "k_Padj"

# create table summarizing rounded raw counts, normalized counts and DGE results
res_k <- merge(k, D_k, by = "ID", all = TRUE)

# call transcripts upregulated, downregulated, or unaffected

res_k$DE <- ifelse(res_k$k_Padj <= 0.05 & res_k$logFC >= 0 & !is.na(res_k$logFC &
    res_k$k_Padj), "Up", ifelse(res_k$k_Padj <= 0.05 & res_k$logFC <= 0 & !is.na(res_k$logFC &
    res_k$k_Padj), "Down", "None"))

# merge with the master dataframe

piRNA_Deg_counts <- merge(piRNA_Deg_counts, res_k, by = "ID")

# generate volcano plot

detags <- rownames(y)[as.logical(de_k)]
```

```
plotSmear(lrt_k, de.tags = detags, cex = 0.2, cex.lab = 1, cex.axis = 1)
```




```
# generate plot of upregulated transcripts expression fold change

Upreg_Types <- c(sum(piRNA_Deg_counts$DE == "Up" & piRNA_Deg_counts$Transcript_Class ==
    "TE"), sum(piRNA_Deg_counts$DE == "Up" & piRNA_Deg_counts$Transcript_Class ==
    "ncRNA"), sum(piRNA_Deg_counts$DE == "Up" & piRNA_Deg_counts$Transcript_Class ==
    "Unchar"), sum(piRNA_Deg_counts$DE == "Up" & piRNA_Deg_counts$Transcript_Class ==
    "Gene"))

lbls <- c("TEs", "ncRNAs", "Unchar", "Genes")
colors = c("red", "blue", "gray", "green")
pct <- round(Upreg_Types/sum(Upreg_Types) * 100)
lbls <- paste(lbls, pct)  # add percents to labels
lbls <- paste(lbls, "%", sep = "")  # ad % to labels
pie(Upreg_Types, labels = lbls, col = colors, main = "hywi RNAi Upregulated Transcript Composition")
```



Somatic Hywi piRNA Mapping Density Ordering

To infer direct targets of Hywi, *hywi* RNAi upregulated transcripts can be described as high and low Hywi piRNA mapping transcripts. "High-mapping"" transcripts are likely to be direct Hywi targets while "low-mapping" transcripts are unlikely to be direct Hywi targets. High Mapping transcripts were defined as those transcripts that were in the top 20% of read counts per kilobase million after combining Colch Hywi Sense and Colch Hywi Antisense reads.

To infer which transcripts were most likely to be involved in the ping-pong cycle in somatic stem cells, transcripts were ordered by Colch Hywi Antisense reads per kilobase million. Transcripts that fall within the top 20% in this category were considered to be putative ping-pong transcripts.

To infer which transcripts may be processed by Hywi phasing in somatic stem cells, transcripts were ordered by Colch Hywi Sense reads per kilobase million. Transcripts that fall within the top 20% in this category were considered to be putative phased transcripts.

The same process was performed for Hyli piRNAs.

```
# Generate percentiles for the different groups

piRNA_Deg_counts$Somatic_Hywi_Perc <- (piRNA_Deg_counts$Colch_Hywi_AS_Counts_RPK +
    piRNA_Deg_counts$Colch_Hywi_S_Counts_RPK)

piRNA_Deg_counts$Somatic_Hywi_Perc <- ecdf(piRNA_Deg_counts$Somatic_Hywi_Perc)(piRNA_Deg_counts$Somatic_

# Repeat for only Colch Hywi Antisense piRNAs
```

```
piRNA_Deg_counts$Somatic_Hywi_Perc_AS <- (piRNA_Deg_counts$Colch_Hywi_AS_Counts_RPK)

piRNA_Deg_counts$Somatic_Hywi_Perc_AS <- ecdf(piRNA_Deg_counts$Somatic_Hywi_Perc_AS)(piRNA_Deg_counts$S

# Repeat for only Colch Hywi Sense piRNAs

piRNA_Deg_counts$Somatic_Hywi_Perc_S <- (piRNA_Deg_counts$Colch_Hywi_S_Counts_RPK)

piRNA_Deg_counts$Somatic_Hywi_Perc_S <- ecdf(piRNA_Deg_counts$Somatic_Hywi_Perc_S)(piRNA_Deg_counts$Soma

# Repeat with Hyli

piRNA_Deg_counts$Somatic_Hyli_Perc <- (piRNA_Deg_counts$Colch_Hyli_AS_Counts_RPK +
    piRNA_Deg_counts$Colch_Hyli_S_Counts_RPK)

piRNA_Deg_counts$Somatic_Hyli_Perc <- ecdf(piRNA_Deg_counts$Somatic_Hyli_Perc)(piRNA_Deg_counts$Somatic_

# Repeat for only Colch Hyli Antisense piRNAs

piRNA_Deg_counts$Somatic_Hyli_Perc_AS <- (piRNA_Deg_counts$Colch_Hyli_AS_Counts_RPK)

piRNA_Deg_counts$Somatic_Hyli_Perc_AS <- ecdf(piRNA_Deg_counts$Somatic_Hyli_Perc_AS)(piRNA_Deg_counts$S

# Repeat for only Colch Hyli Sense piRNAs

piRNA_Deg_counts$Somatic_Hyli_Perc_S <- (piRNA_Deg_counts$Colch_Hyli_S_Counts_RPK)

piRNA_Deg_counts$Somatic_Hyli_Perc_S <- ecdf(piRNA_Deg_counts$Somatic_Hyli_Perc_S)(piRNA_Deg_counts$Soma

# write table that summarizes data write.table(piRNA_Deg_counts, file =
# 'objects/Annotated_piRNA_Degradome_DGE_Count_Matrix.txt', sep = '\t')
```

GO-Term Enrichment Analysis

GO-term enrichment analysis was performed on upregulated transcripts against the entire transcriptome to investigate a functional response to somatic *hywi* knockdown.

GO-term enrichment analysis was performed using goatools v0.6.10 using the script: goatools_GO_enrichment.pl

```
# Load GO annotation results

GO_table <- read.table("objects/GO_upreg_trans_full_ref.txt", header = T, sep = "\t")

# Take enriched biological processes

GO_table_sub <- subset(GO_table, p_bonferroni <= 0.05 & NS == "BP")

# Include GO accession number, GO term, ratio in study, ratio in population,
# bonferroni corrected p-value, and transcript IDs

GO_table_sub <- GO_table_sub[, c(1, 4:6, 10, 14)]
colnames(GO_table_sub) <- c("GO", "GO_Term", "Study", "Pop", "p_val", "ID")
```

```
# Print table

kable(GO_table_sub, format = "markdown", padding = 100)
```

GO	GO_Term	Study	Pop	p_val	ID
...GO:0006952	defense response	26/441	753/38747	0.0142	t11117aep, t12198aep, t16424aep, t17178aep, t17750aep, t21013aep, t21682aep, t22133aep, t24687aep, t32280aep, t33020aep, t34385aep, t34424aep, t34475aep, t35573aep, t35608aep, t35837aep, t38672aep, t38673aep, t5914aep, t6387aep, t7326aep, t7388aep, t8582aep, t8645aep, t8946aep
....GO:0051899	membrane depolarization	7/441	46/38747	0.0170	t17803aep, t18338aep, t24938aep, t25020aep, t526aep, t527aep, t9936aep
.......GO:2000051	negative regulation of non-canonical Wnt signaling pathway	4/441	8/38747	0.0221	t29674aep, t30176aep, t33022aep, t8142aep
....GO:0048440	carpel development	3/441	3/38747	0.0290	t35573aep, t38672aep, t38673aep
...GO:0045087	innate immune response	14/441	258/38747	0.0370	t11117aep, t17178aep, t22133aep, t24687aep, t34385aep, t34424aep, t35573aep, t35608aep, t35837aep, t38672aep, t38673aep, t6387aep, t7326aep, t8946aep
.......GO:0031050	dsRNA processing	5/441	19/38747	0.0377	t30770aep, t35488aep, t38672aep, t38673aep, t5914aep
......GO:0042108	positive regulation of cytokine biosynthetic process	5/441	20/38747	0.0498	t21682aep, t22133aep, t24687aep, t34424aep, t8645aep

Software versions

This document was computed on Fri Jan 10 10:35:54 2020 with the following R package versions.

```
R version 3.5.3 (2019-03-11)
Platform: x86_64-apple-darwin15.6.0 (64-bit)
Running under: macOS Mojave 10.14.5

Matrix products: default
BLAS: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/3.5/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] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] ggplot2_3.2.0 xtable_1.8-3  edgeR_3.22.5  limma_3.38.3  knitr_1.22

loaded via a namespace (and not attached):
 [1] Rcpp_1.0.1       magrittr_1.5     splines_3.5.3    tidyselect_0.2.5
 [5] munsell_0.5.0    colorspace_1.4-1 lattice_0.20-38  R6_2.4.0
 [9] rlang_0.4.0      highr_0.7        dplyr_0.8.3      stringr_1.4.0
[13] tools_3.5.3      grid_3.5.3       gtable_0.3.0     xfun_0.5
[17] withr_2.1.2      htmltools_0.3.6  assertthat_0.2.1 yaml_2.2.0
[21] lazyeval_0.2.2   digest_0.6.20    tibble_2.1.3     crayon_1.3.4
[25] purrr_0.3.2      formatR_1.7      glue_1.3.1       evaluate_0.13
[29] rmarkdown_1.12   stringi_1.4.3    compiler_3.5.3   pillar_1.4.2
[33] scales_1.0.0     locfit_1.5-9.1   pkgconfig_2.0.2
```

3 Ping Pong Analysis

Bryan Teefy

12/20/2019

Ping-Pong Hits

PIWI targets are often degraded with a distinctive "ping-pong signature" consisting of a 10 bp overlap between the 5' ends of antisense-mapped piRNAs and sense-mapped piRNAs/degradome reads. Transcripts that had a ping-pong signature comprised of at least 10 of the contributing species (antisense piRNA, sense piRNA and degradome read) in the correct orientation were deemed "ping-pong hits". Based on PIWI protein homology, a "canonical" ping-pong hit occurs between antisense-mapped Hywi piRNAs and sense-mapped Hyli piRNAs while an "inverse" ping-pong hit occurs between an antisense-mapped Hyli piRNA and a sense-mapped Hywi piRNA.

2047 transcripts have a "canonincal" ping-pong hit in whole animals compared to only 709 transcripts with an "inverse" ping-pong hit in whole animals. Likewise, 254 transcripts have a "canonical" ping-pong hit in epithelial animals while 74 transcripts have "inverse" ping-pong hits in epithelial animals. This suggests that in *Hydra*, canonical ping-pong is more common in either lineage.

To identify transcripts with "ping-pong" hits, we used the following approach:

piRNA and Degradome BAM files generated from the Bowtie mapping strategy in RMD1 were converted to BED files using the script: bam_to_bed.sh.

The script "grouping.sh" retains only those transcripts that have at least 10 piRNA/degradome reads beginning at a particular position such that the depth criteria for calling a ping-pong hit can be met. The script outputs transcript ID, start position of a piRNA/degradome read, and 10 bps from the start of the piRNA/degradome read. Adding 10 bps from the start position allows for the subsequent overlap_BT.perl script to find reads that overlap by the specified 10 bp length.
grouping.sh uses: "group_reads_sense.perl" and "group_reads_antisense.perl"

The script "overlap.sh" generates a matrix containing ping-pong hit coordinates.
"overlap.sh" makes uses: "overlap_BT.perl"

"overlap.sh" output files were converted to .txt files, merged in R, and used to generate TRUE/FALSE statements in reference to whether a transcript has a Whole Animal or a Epithelial Animal canonical or inverse ping-pong hit.

The resultant table is: "Ping_Pong_Matrix_Bowtie.txt"

Load the Necessary Files

```
#Load normalized piRNA and degradome counts, expression data, DGE data from RMD2.

Read_Counts_Master_DF <- read.table("objects/Annotated_piRNA_Degradome_DGE_Count_Matrix.txt", sep = "\t

#Load binary matrix of ping-pong hits per transcript.

Ping_Pong_Matrix <- read.table("objects/Ping_Pong_Matrix.txt")
```

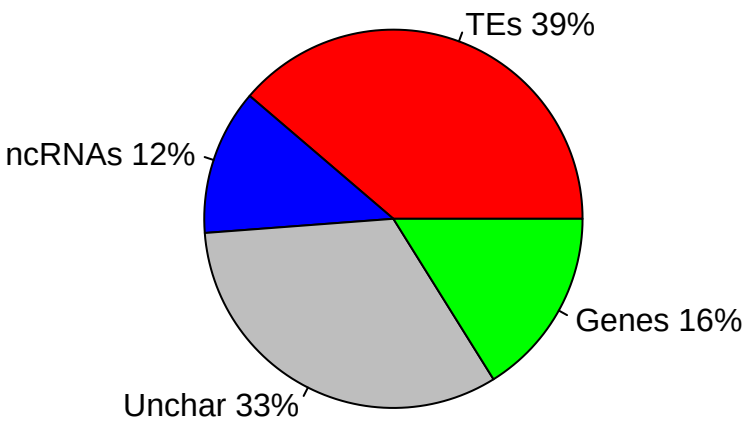
Load Necessary Packages

```
###load packages###

library(ggplot2)

library(reshape2)

library(ggpubr)
```

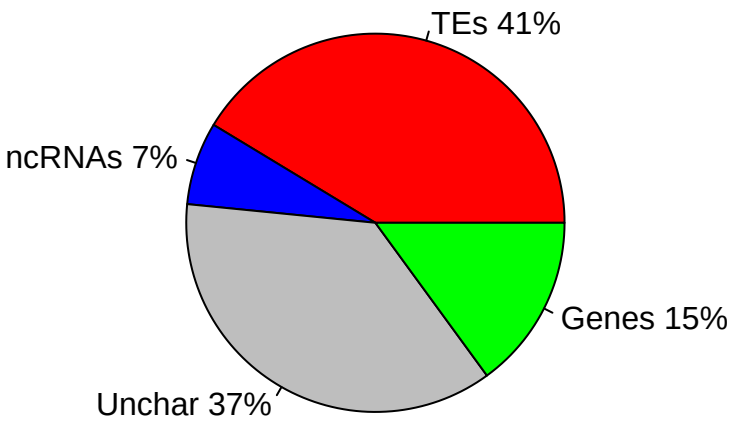
Visualizing Ping-Pong Hits

To visualize ping-pong hit distribution, we generated pie charts consisting of all transcripts that have at least one ping-pong hit and plot the output by transcript class for 1) Whole Animals and 2) Epithelial Animals.

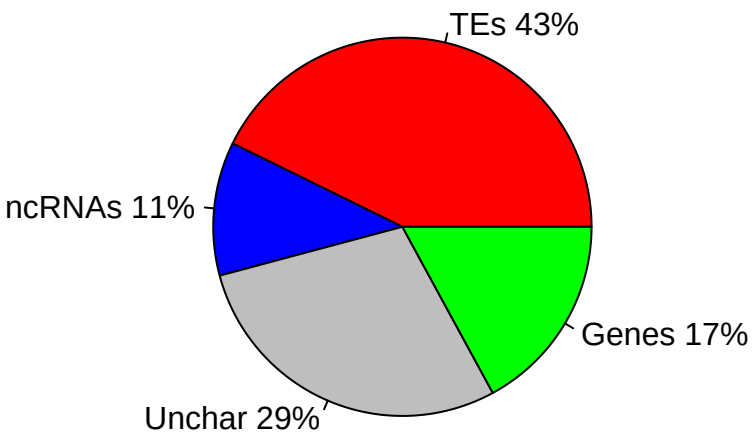
```
Read_Counts_Master_DF <- merge(Read_Counts_Master_DF, Ping_Pong_Matrix, by = "ID", all = T)

#Retrieve hits

WT_ping_hits <- c(sum(Read_Counts_Master_DF$Whole_Ping_Pong  & Read_Counts_Master_DF$Transcript_Class ==

Colch_ping_hits <- c(sum(Read_Counts_Master_DF$Epi_Ping_Pong  & Read_Counts_Master_DF$Transcript_Class =

#Create pie chart with percentages

#Whole Animal Canonical
lbls <- c("TEs", "ncRNAs", "Unchar", "Genes")
colors = c("red", "blue", "gray", "green")
pct <- round(WT_ping_hits/sum(WT_ping_hits)*100)
lbls <- paste(lbls, pct) # add percents to labels
lbls <- paste(lbls,"%",sep="") # ad % to labels
pie(WT_ping_hits,labels = lbls, col=colors,
    main="Whole Animal Ping-Pong Hits")
```



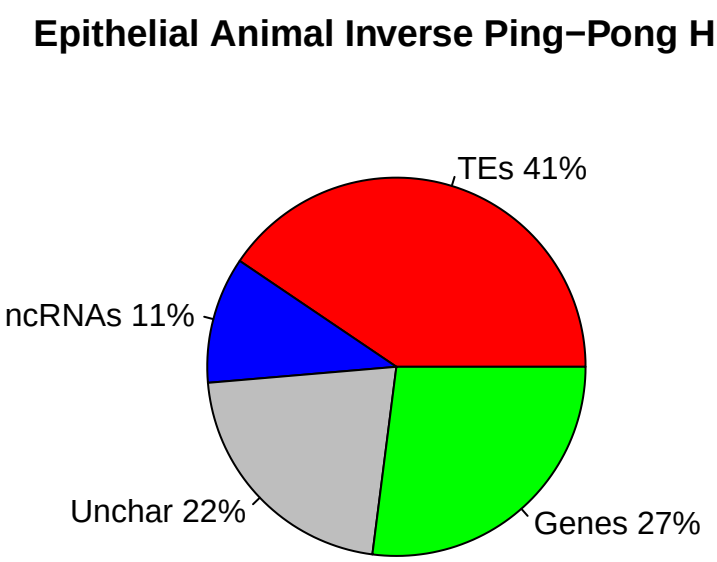

```
#Epithelial Animal Canonical
lbls <- c("TEs", "ncRNAs", "Unchar", "Genes")
colors = c("red", "blue", "gray", "green")
pct <- round(Colch_ping_hits/sum(Colch_ping_hits)*100)
lbls <- paste(lbls, pct) # add percents to labels
lbls <- paste(lbls,"%",sep="") # ad % to labels
pie(Colch_ping_hits,labels = lbls, col=colors,
    main="Epithelial Animal Ping-Pong Hits")
```




```
#Repeat for inverse ping-pong

WT_inv_ping_hits <- c(sum(Read_Counts_Master_DF$Whole_Ping_Pong_Inverse  & Read_Counts_Master_DF$Transc

Colch_inv_ping_hits <- c(sum(Read_Counts_Master_DF$Epi_Ping_Pong_Inverse  & Read_Counts_Master_DF$Trans


#Whole Animal Inverse
lbls <- c("TEs", "ncRNAs", "Unchar", "Genes")
colors = c("red", "blue", "gray", "green")
pct <- round(WT_inv_ping_hits/sum(WT_inv_ping_hits)*100)
lbls <- paste(lbls, pct) # add percents to labels
lbls <- paste(lbls,"%",sep="") # ad % to labels
pie(WT_inv_ping_hits,labels = lbls, col=colors,
    main="Whole Animal Inverse Ping-Pong Hits")
```




```
#Epithelial Animal Inverse
lbls <- c("TEs", "ncRNAs", "Unchar", "Genes")
colors = c("red", "blue", "gray", "green")
pct <- round(Colch_inv_ping_hits/sum(Colch_inv_ping_hits)*100)
lbls <- paste(lbls, pct) # add percents to labels
lbls <- paste(lbls,"%",sep="") # ad % to labels
pie(Colch_inv_ping_hits,labels = lbls, col=colors,
    main="Epithelial Animal Inverse Ping-Pong Hits")
```



piRNA Positional Overlap Frequency Interrogation

Ping-pong processing of transcripts is known to occur in *Hydra* but whether this type of processing is active in somatic stem cells remains unknown. piRNAs from Whole Animals and Epithelial Animals were used to address this problem by generating piRNA overlap frequency plots. Ping-pong processing should consist of a preponderance of 10 bp overlaps between antisense- and sense-mapped piRNAs.

Positional information was extracted from piRNA BAM files and exported as a .txt file using the script: get_position.sh This script uses: "get_rep_piRNA_sense_BT.perl" and "get_rep_piRNA_antisense_BT.perl".

txt files were converted into BED files using the script: "Overlap_bed_ping.sh."

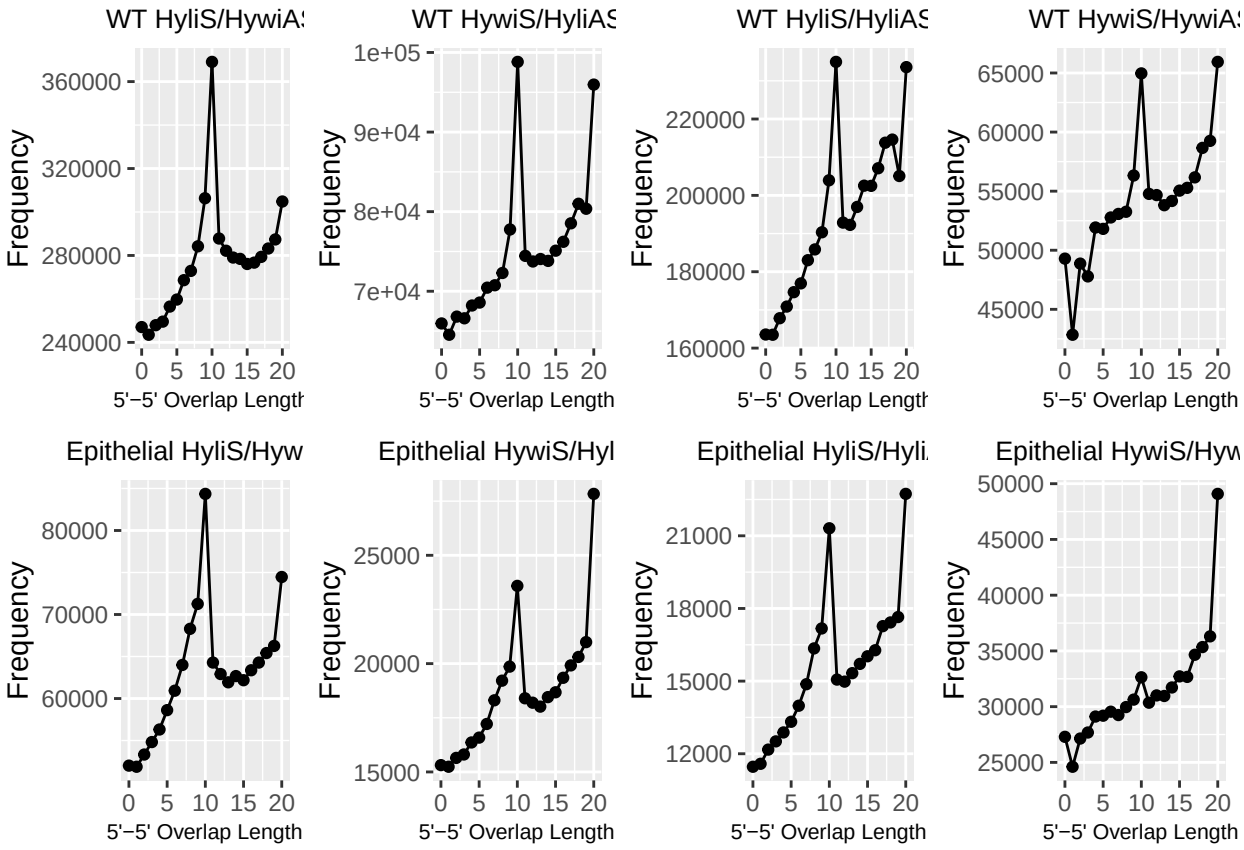
BED files containing piRNA overlap positions and overlap length between pairs of piRNAs (i.e. Hywi antisense/Hyli sense) were generated using the script "windowbed_ping.sh."

The output of "windowbed_ping.sh." are BED files consisting of rows indexing a piRNA overlap event (ex. an overlap event between piRNA_1 and piRNA_2) within a 30 bp window.
The file contains has 11 columns:
1) Transcript on which piRNA_1 is mapped 2) Start position of piRNA_1 3) End position of piRNA_1 4) Sequence of piRNA_1 5) Copy number of piRNA_1
6) Transcript on which piRNA_2 is mapped 7) Start position of piRNA_2 8) End position of piRNA_2 9) Sequence of piRNA_2 10) Copy number of piRNA_2 11) 5'-5' piRNA overlap length

Overlap lengths were computed and compilied into a matrix entitled "Ping_Pong_Overlap_Matrix" using the R script, "Ping_Pong_Matrix_Generation.R" and run using the script, "run_Ping_Pong_Matrix_Generation.sh"

piRNA overlaps were queried in 0 to 20 base pair overlap window as in Gainetdinov *et al.*, 2018.

```r
#Load Matrices of piRNA overlap events

Ping_Pong_Overlap_Matrix <- read.table("objects/Ping_Pong_Overlap_Matrix")

#Plot individual Overlap Frequency Plots

a <- ggplot(data=Ping_Pong_Overlap_Matrix, aes(x=Overlap_Length, y=WT_HyliS_HywiAS, group = 1)) +
  geom_line(stat="identity") + xlab("5'-5' Overlap Length") + ylab("Frequency") + ggtitle("WT HyliS/Hyw
  geom_point()

b <- ggplot(data=Ping_Pong_Overlap_Matrix, aes(x=Overlap_Length, y=WT_HywiS_HyliAS, group = 1)) +
  geom_line(stat="identity") + xlab("5'-5' Overlap Length") + ylab("Frequency") + ggtitle("WT HywiS/Hyli
  geom_point()

c <- ggplot(data=Ping_Pong_Overlap_Matrix, aes(x=Overlap_Length, y=WT_HyliS_HyliAS, group = 1)) +
  geom_line(stat="identity") + xlab("5'-5' Overlap Length") + ylab("Frequency") + ggtitle("WT HyliS/Hyli
  geom_point()

d <- ggplot(data=Ping_Pong_Overlap_Matrix, aes(x=Overlap_Length, y=WT_HywiS_HywiAS, group = 1)) +
  geom_line(stat="identity") + xlab("5'-5' Overlap Length") + ylab("Frequency") + ggtitle("WT HywiS/Hyw
  geom_point()

e <- ggplot(data=Ping_Pong_Overlap_Matrix, aes(x=Overlap_Length, y=Colch_HyliS_HywiAS, group = 1)) +
  geom_line(stat="identity") + xlab("5'-5' Overlap Length") + ylab("Frequency") + ggtitle("Epithelial Hy
  geom_point()

f <- ggplot(data=Ping_Pong_Overlap_Matrix, aes(x=Overlap_Length, y=Colch_HywiS_HyliAS, group = 1)) +
  geom_line(stat="identity") + xlab("5'-5' Overlap Length") + ylab("Frequency") + ggtitle("Epithelial Hy
  geom_point()

g <- ggplot(data=Ping_Pong_Overlap_Matrix, aes(x=Overlap_Length, y=Colch_HyliS_HyliAS, group = 1)) +
  geom_line(stat="identity") + xlab("5'-5' Overlap Length") + ylab("Frequency") + ggtitle("Epithelial Hy
  geom_point()

h <- ggplot(data=Ping_Pong_Overlap_Matrix, aes(x=Overlap_Length, y=Colch_HywiS_HywiAS, group = 1)) +
  geom_line(stat="identity") + xlab("5'-5' Overlap Length") + ylab("Frequency") + ggtitle("Epithelial Hy
  geom_point()

#Arrange ping-pong frequency plots

ggarrange(a,b,c,d,e,f,g,h, ncol = 4, nrow = 2)
```

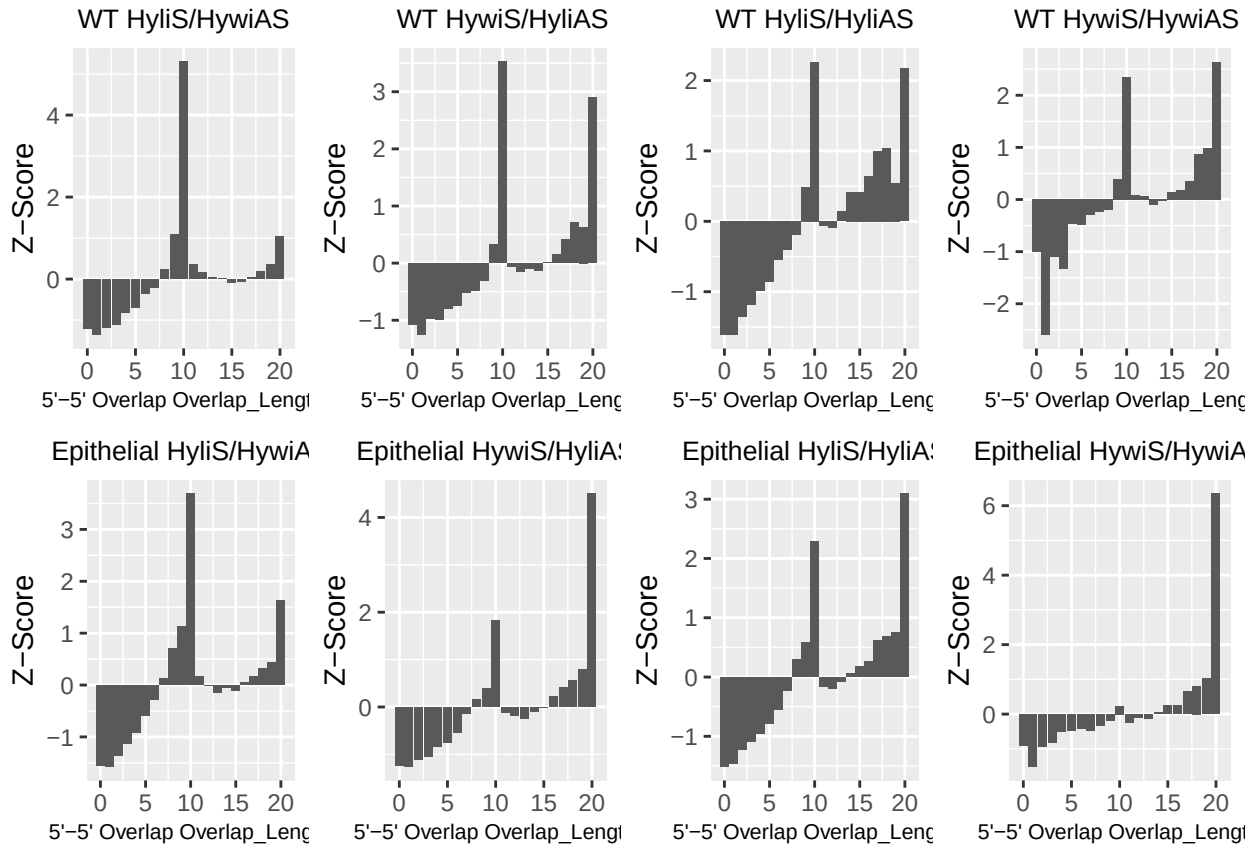


Z10 scores

To quantify the strength of the ping-pong signal, we generated Z-scores for the 5'-5' overlap frequency data of piRNAs used in the previous analysis. Z-scores were taken from a range of 0 to 20 base pairs. Z10 scores are reported in the text.

```
#Import 5'-5' overlap frequency data and remove length information

Ping_Pong_No_Length <- Ping_Pong_Overlap_Matrix[,c(2:9)]

#Store empty vector for results

results <- vector("list",length(ncol(Ping_Pong_No_Length)))

#Generate nested for-loop to compute Z-score at every overlap length for each ping-pong pair (i.e. Hywi

for (j in 1:ncol(Ping_Pong_No_Length)) {
  col.use <- Ping_Pong_No_Length[,j]
  res <- numeric(length = length(col.use))
  n <- 0
  for (i in col.use) {
    message(i)
    notx <- col.use[!(col.use == i)]
    z <- ((i - mean(notx))/(sd(notx)))
    message(z)
```

```
    message(n)
    n <- n + 1
    message(n)
    res[n] <- z
    obj <- as.data.frame(res)
  }
  results[[j]] <- res
}

#Bind names and overlap lengths to Z-scores

results.df <- do.call(cbind,results)

final_res <- cbind(Ping_Pong_Overlap_Matrix$Overlap_Length, results.df)

Col_Names <- c("Overlap_Length", "WT_HywiS_HywiAS","WT_HywiS_HyliAS", "WT_HyliS_HyliAS", "WT_HyliS_Hywi

colnames(final_res) <- paste(Col_Names, sep = "")

final_res <- as.data.frame(final_res)

#Plot individual Z-scores

a<-ggplot(data=final_res, aes(x=Overlap_Length, y=WT_HyliS_HywiAS)) +
  geom_bar(stat="identity") + xlab("5'-5' Overlap Overlap_Length") + ylab("Z-Score") + ggtitle("WT Hyli

b<-ggplot(data=final_res, aes(x=Overlap_Length, y=WT_HywiS_HyliAS)) +
  geom_bar(stat="identity") + xlab("5'-5' Overlap Overlap_Length") + ylab("Z-Score") + ggtitle("WT Hywi

c<-ggplot(data=final_res, aes(x=Overlap_Length, y=WT_HyliS_HyliAS)) +
  geom_bar(stat="identity") + xlab("5'-5' Overlap Overlap_Length") + ylab("Z-Score") + ggtitle("WT Hyli

d<-ggplot(data=final_res, aes(x=Overlap_Length, y=WT_HywiS_HywiAS)) +
  geom_bar(stat="identity") + xlab("5'-5' Overlap Overlap_Length") + ylab("Z-Score") + ggtitle("WT Hywi

e<-ggplot(data=final_res, aes(x=Overlap_Length, y=Colch_HyliS_HywiAS)) +
  geom_bar(stat="identity") + xlab("5'-5' Overlap Overlap_Length") + ylab("Z-Score") + ggtitle("Epitheli

f<-ggplot(data=final_res, aes(x=Overlap_Length, y=Colch_HywiS_HyliAS)) +
  geom_bar(stat="identity") + xlab("5'-5' Overlap Overlap_Length") + ylab("Z-Score") + ggtitle("Epitheli

g<-ggplot(data=final_res, aes(x=Overlap_Length, y=Colch_HyliS_HyliAS)) +
  geom_bar(stat="identity") + xlab("5'-5' Overlap Overlap_Length") + ylab("Z-Score") + ggtitle("Epitheli

h<-ggplot(data=final_res, aes(x=Overlap_Length, y=Colch_HywiS_HywiAS)) +
  geom_bar(stat="identity") + xlab("5'-5' Overlap Overlap_Length") + ylab("Z-Score") + ggtitle("Epitheli

#Arrange Z-score plots

ggarrange(a,b,c,d,e,f,g,h, ncol = 4, nrow = 2)
```



Phasing

Phasing is a conserved mechanism of piRNA biogenesis in which an RNA molecule is processively converted into a piRNA. If phasing occurs in a manner such that there is no piRNA trimming, there should be no distance between the 3' and 5' ends of adjacent, mature piRNAs. Thus the Z0 score, the distance between 3' and 5' piRNA ends, should be above 1.96 (p-value of 0.05) if phasing is occuring. Z-scores were calculated from a range of -10 to 50 base pair overlaps.

BED files containing piRNA overlap positions and overlap length between pairs of piRNAs from the same group (i.e. WT Hywi sense/WT Hywi sense) were generated using the script "windowBED_phasing.sh" using the output of "Overlap_bed_ping.sh."

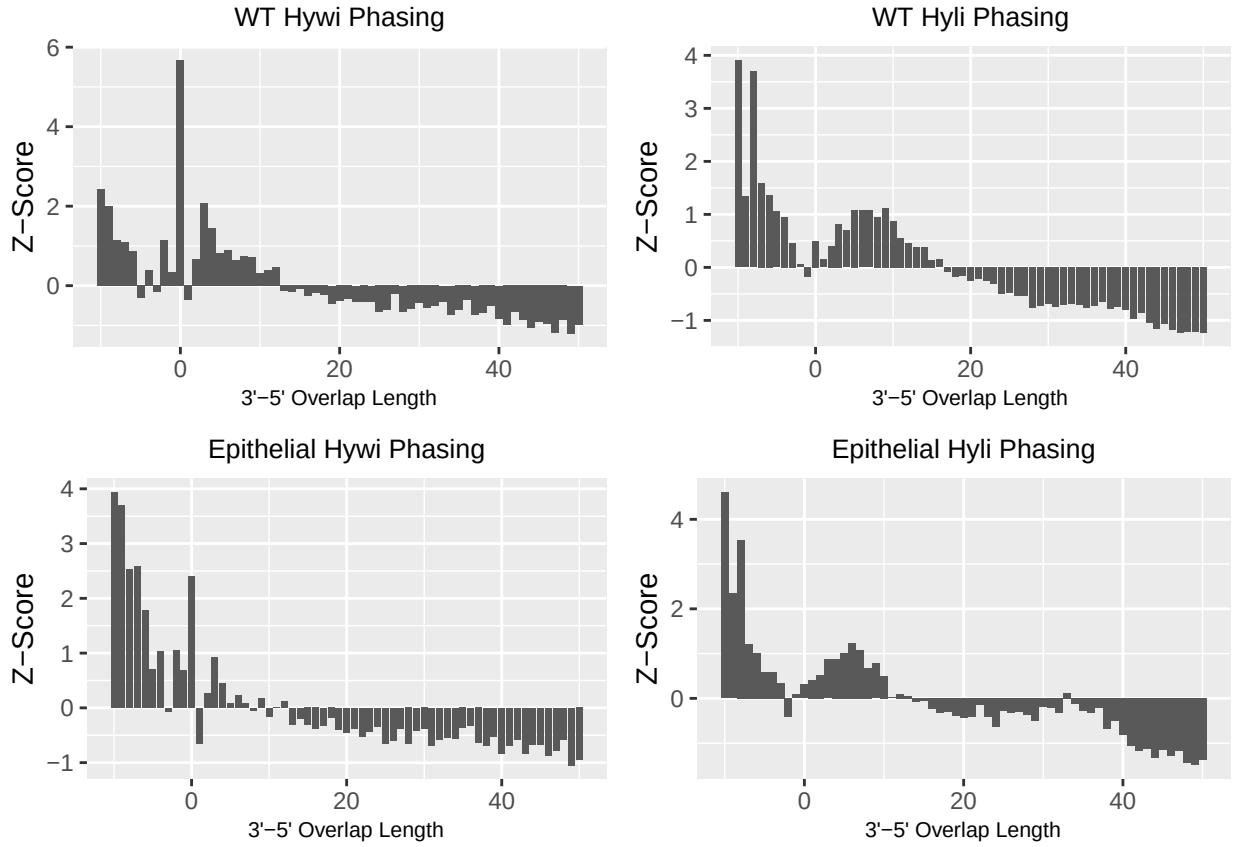
The output of windowBED_phasing.sh are BED files consisting of rows indexing a piRNA overlap event (ex. an overlap event between piRNA_1 and piRNA_2) within a 200 bp window.
The file contains has 11 columns:
1) Transcript on which piRNA_1 is mapped 2) Start position of piRNA_1 3) End position of piRNA_1 4) Sequence of piRNA_1 5) Copy number of piRNA_1
6) Transcript on which piRNA_2 is mapped 7) Start position of piRNA_2 8) End position of piRNA_2 9) Sequence of piRNA_2 10) Copy number of piRNA_2 11) 5'-5' piRNA overlap length

Overlap lengths were computed and compiled into a matrix entitled "Phasing_Overlap_Matrix" using the R script, "Phasing.R" and run using the script, "run_Phasing.sh". Importantly, self-referential piRNAs are ommited such that the 3'-5' distance between piRNAs that pair with themselves was not calculated.

Below, Z-scores are calculated for 3'-5' distances between piRNAs. 3'-5' distances were queried in -10 to 50 base pair window as in Gainetdinov *et al.*, 2018.

```
#Import Phasing Matrix

Phasing_Overlap_Matrix <- read.table("objects/Phasing_Overlap_Matrix")

#Remove Length Information

Phasing_No_Length <- Phasing_Overlap_Matrix[,c(2:5)]

#Store empty vector for results

results <- vector("list",length(ncol(Phasing_No_Length)))

#Generate nested for-loop to compute Z-score at every overlap length

for (j in 1:ncol(Phasing_No_Length)) {
  col.use <- Phasing_No_Length[,j]
  res <- numeric(length = length(col.use))
  n <- 0
  for (i in col.use) {
    message(i)
    notx <- col.use[!(col.use == i)]
    z <- ((i - mean(notx))/(sd(notx)))
    message(z)
    message(n)
    n <- n + 1
    message(n)
    res[n] <- z
    obj <- as.data.frame(res)
  }
  results[[j]] <- res
}

#Bind names and overlap lengths to Z-scores

results.df <- do.call(cbind,results)

final_res <- cbind((Phasing_Overlap_Matrix$Length), results.df)

Col_Names_Phasing <- c("Length", "WT_Hywi_S","WT_Hyli_S", "Colch_Hywi_S", "Colch_Hyli_S")

colnames(final_res) <- paste(Col_Names_Phasing, sep = "")

final_res <- as.data.frame(final_res)

#Plot individual Z-scores

a <- ggplot(data=final_res, aes(x=Length, y=WT_Hywi_S)) +
  geom_bar(stat="identity") + xlab("3'-5' Overlap Length") + ylab("Z-Score") + ggtitle("WT Hywi Phasing

b <- ggplot(data=final_res, aes(x=Length, y=WT_Hyli_S)) +
  geom_bar(stat="identity") + xlab("3'-5' Overlap Length") + ylab("Z-Score") + ggtitle("WT Hyli Phasing

c <- ggplot(data=final_res, aes(x=Length, y=Colch_Hywi_S)) +
```

```
  geom_bar(stat="identity") + xlab("3'-5' Overlap Length") + ylab("Z-Score") + ggtitle("Epithelial Hywi

d <- ggplot(data=final_res, aes(x=Length, y=Colch_Hyli_S)) +
  geom_bar(stat="identity") + xlab("3'-5' Overlap Length") + ylab("Z-Score") + ggtitle("Epithelial Hyli

#Arrange

ggarrange(a,b,c,d, ncol = 2, nrow = 2)
```




```
#Write table

#write.csv(Read_Counts_Master_DF, file = "objects/Table_S1.csv")
```

Software versions

This document was computed on Fri Jan 10 10:43:43 2020 with the following R package versions.

```
R version 3.5.3 (2019-03-11)
Platform: x86_64-apple-darwin15.6.0 (64-bit)
Running under: macOS Mojave 10.14.5

Matrix products: default
BLAS: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/3.5/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] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] ggpubr_0.2.4   magrittr_1.5   reshape2_1.4.3 ggplot2_3.2.0

loaded via a namespace (and not attached):
 [1] Rcpp_1.0.1       knitr_1.22       cowplot_0.9.4    tidyselect_0.2.5
 [5] munsell_0.5.0    colorspace_1.4-1 R6_2.4.0         rlang_0.4.0
 [9] plyr_1.8.4       stringr_1.4.0    dplyr_0.8.3      tools_3.5.3
[13] grid_3.5.3       gtable_0.3.0     xfun_0.5         withr_2.1.2
[17] htmltools_0.3.6  yaml_2.2.0       lazyeval_0.2.2   digest_0.6.20
[21] assertthat_0.2.1 tibble_2.1.3     ggsignif_0.5.0   crayon_1.3.4
[25] purrr_0.3.2      glue_1.3.1       evaluate_0.13    rmarkdown_1.12
[29] labeling_0.3     stringi_1.4.3    compiler_3.5.3   pillar_1.4.2
[33] scales_1.0.0     pkgconfig_2.0.2
```

4 Lineage-sorted piRNA Count Generation

Bryan Teefy

12/20/2019

Load Required Libraries

```
library(dplyr)
library(reshape2)
library(ggplot2)
library(ggpubr)
library(VennDiagram)
```

Set Copy Number Thresholds for small RNA sequences

To determine which piRNA sequences might be unique to each of the three cell lineages in *Hydra*, we contrasted our Whole Animal Hywi and Hyli piRNA datasets with FAC-sorted lineage-specific small RNA libraries (Juliano *et al.*, 2014).

To control for the abundance of common versus exclusive piRNAs in our downstream analysis, we initially filtered out the lower copy small RNAs from each dataset. The threshold chosen for each dataset was 4 copies based on count frequency distributions visualized below.

First, small RNA files were mapped to the *Hydra* transcriptome using Bowtie v1.1.2 using the shell script, "sRNA_mapping.sh."
The options "–best –strata -k 1" were used because at this stage, mapping all possible hits was not yet necessary.

Copy number of unique piRNA sequences were quantified from BAM files generated from small RNA mapping using the script "get_pos_small_RNA.sh." This script made use of the perl scripts "get_rep_piRNA_sense_BT.perl" and "get_rep_piRNA_antisense_BT.perl".

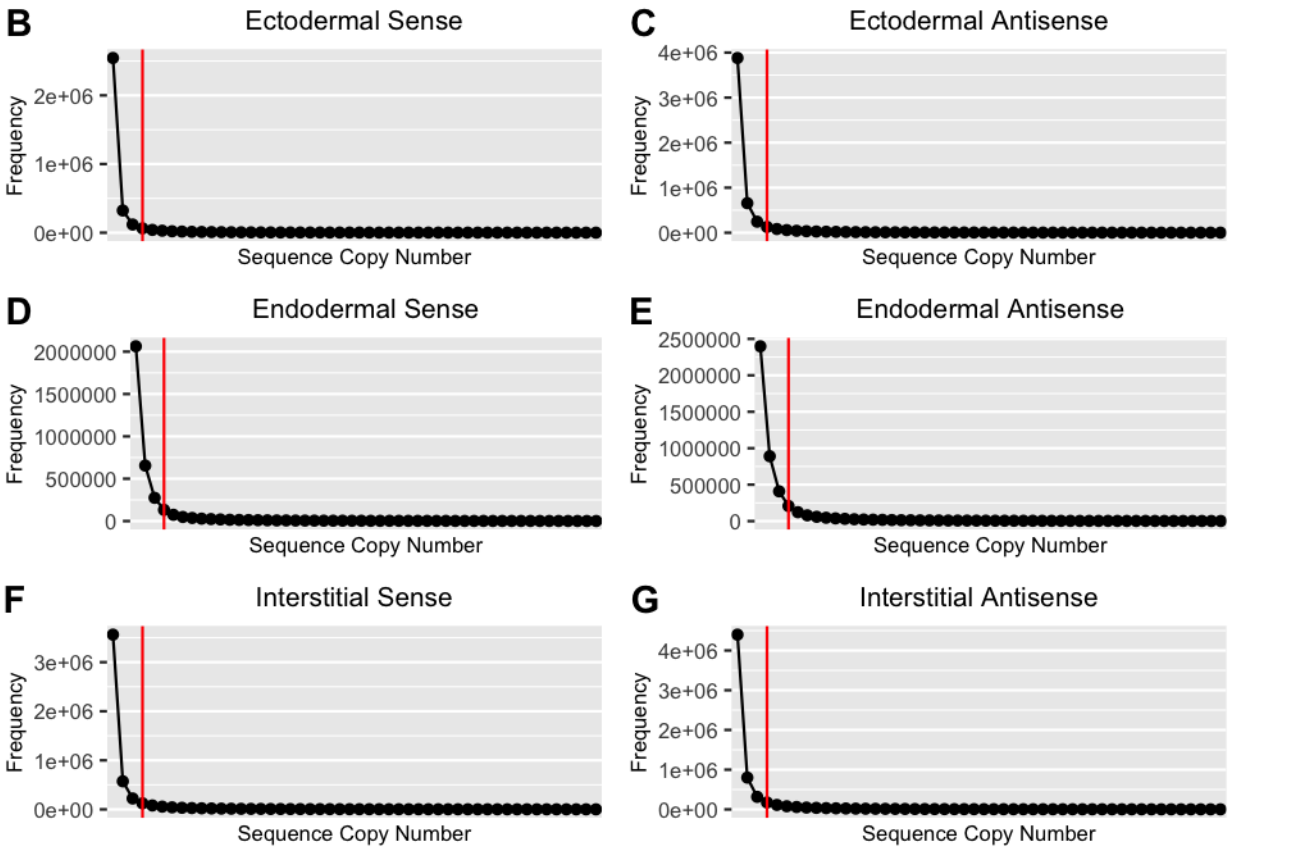
Plots of sequence copy number frequency were generated using the R script, "plot_gen.R" and run with the script, "run_plot_gen.sh."

Plots for each lineage and mapping orientation are visualized below.

```
# Sense Ecto

Ect_S_plot <- read.table("objects/Ect_S_plot")
A <- ggplot(data = Ect_S_plot, aes(x = Var1, y = Freq, group = 1)) + geom_line() +
    geom_point()
rm(Ect_S)
rm(Ect_S_plot)

# Endo

End_S_plot <- read.table("objects/End_S_plot")
B <- ggplot(data = End_S_plot, aes(x = Var1, y = Freq, group = 1)) + geom_line() +
    geom_point()
rm(End_S)
rm(End_S_plot)
```

```
# Int

Int_S_plot <- read.table("objects/Int_S_plot")
C <- ggplot(data = Int_S_plot, aes(x = Var1, y = Freq, group = 1)) + geom_line() +
    geom_point()
rm(Int_S)
rm(Int_S_plot)

# Antisense Ecto

Ect_AS_plot <- read.table("objects/Ect_AS_plot")
D <- ggplot(data = Ect_AS_plot, aes(x = Var1, y = Freq, group = 1)) + geom_line() +
    geom_point()
rm(Ect_AS)
rm(Ect_AS_plot)

# Endo

End_AS_plot <- read.table("objects/End_AS_plot")
E <- ggplot(data = End_AS_plot, aes(x = Var1, y = Freq, group = 1)) + geom_line() +
    geom_point()
rm(End_AS)
rm(End_AS_plot)

# Int

Int_AS_plot <- read.table("objects/Int_AS_plot")
G <- ggplot(data = Int_AS_plot, aes(x = Var1, y = Freq, group = 1)) + geom_line() +
    geom_point()

rm(Int_AS)
rm(Int_AS_plot)


# Plot

A <- A + geom_vline(xintercept = 4, color = "red") + scale_x_discrete(breaks = seq(0,
    50, 5)) + ggtitle("Ectodermal Sense") + theme(plot.title = element_text(hjust = 0.5,
    size = 10), axis.title.x = element_text(size = 8), axis.title.y = element_text(size = 8),
    axis.text.y = element_text(size = 8), axis.text.x = element_text(size = 8)) +
    xlab("Sequence Copy Number") + ylab("Frequency")

B <- B + geom_vline(xintercept = 4, color = "red") + scale_x_discrete(breaks = seq(0,
    50, 5)) + ggtitle("Endodermal Sense") + theme(plot.title = element_text(hjust = 0.5,
    size = 10), axis.title.x = element_text(size = 8), axis.title.y = element_text(size = 8),
    axis.text.y = element_text(size = 8), axis.text.x = element_text(size = 8)) +
    xlab("Sequence Copy Number") + ylab("Frequency")

C <- C + geom_vline(xintercept = 4, color = "red") + scale_x_discrete(breaks = seq(0,
    50, 5)) + ggtitle("Interstitial Sense") + theme(plot.title = element_text(hjust = 0.5,
    size = 10), axis.title.x = element_text(size = 8), axis.title.y = element_text(size = 8),
    axis.text.y = element_text(size = 8), axis.text.x = element_text(size = 8)) +
    xlab("Sequence Copy Number") + ylab("Frequency")
```

```
D <- D + geom_vline(xintercept = 4, color = "red") + scale_x_discrete(breaks = seq(0,
    50, 5)) + ggtitle("Ectodermal Antisense") + theme(plot.title = element_text(hjust = 0.5,
    size = 10), axis.title.x = element_text(size = 8), axis.title.y = element_text(size = 8),
    axis.text.y = element_text(size = 8), axis.text.x = element_text(size = 8)) +
    xlab("Sequence Copy Number") + ylab("Frequency")


E <- E + geom_vline(xintercept = 4, color = "red") + scale_x_discrete(breaks = seq(0,
    50, 5)) + ggtitle("Endodermal Antisense") + theme(plot.title = element_text(hjust = 0.5,
    size = 10), axis.title.x = element_text(size = 8), axis.title.y = element_text(size = 8),
    axis.text.y = element_text(size = 8), axis.text.x = element_text(size = 8)) +
    xlab("Sequence Copy Number") + ylab("Frequency")


G <- G + geom_vline(xintercept = 4, color = "red") + scale_x_discrete(breaks = seq(0,
    50, 5)) + ggtitle("Interstitial Antisense") + theme(plot.title = element_text(hjust = 0.5,
    size = 10), axis.title.x = element_text(size = 8), axis.title.y = element_text(size = 8),
    axis.text.y = element_text(size = 8), axis.text.x = element_text(size = 8)) +
    xlab("Sequence Copy Number") + ylab("Frequency")

# Arrange
ggarrange(A, D, B, E, C, G, ncol = 2, nrow = 3, labels = c("B", "C", "D", "E", "F",
    "G"))
```



Generate Lineage-sorted piRNA Libraries

As stated above, to control for the abundance of common versus exclusive piRNAs in our downstream analysis, we removed small RNAs sequences from each library with a copy number of fewer than 4. This was done using the R script, "remove_low_copy_RNAs.R" and run using the script, "run_remove_low_copy_RNAs.sh."

In order to more easily contrast piRNA and small RNA sequences, the trimmed piRNA fastq libraries and the trimmed small RNA fastq files were converted to dataframes that retained 1) sequences and 2) fastq headers using the script, "RNA_Table_Generation.R" and run using the script, "run_RNA_Table_Generation.sh."

The small RNA library dataframes were then contrasted with the small RNA sequences with 4 or greater copies using the script, "filtering_small_RNAs_under_four.R" and run with the script, "run_filtering_small_RNAs_under_four.sh." This generated small RNA dataframes consisting of only those small RNAs whose copy numbers were 4 or greater.

Next, piRNA and small RNA dataframes were contrasted such that only matching sequences were retained. Crucially, sequence copy number was reflective of the small RNA library so lineage-specific piRNA abundancies were reflected. This was done using the script, "piRNA_contrast_with_over_four_small_RNAs.R" and run using the script, "run_piRNA_contrast_with_over_four_small_RNAs.sh."

To map the resultant lineage-specific piRNA libraries, they were converted into FASTA files using the script, "make_fasta_from_piRNA_sRNA_cross.R" and run using the script, "run_make_fasta_from_piRNA_sRNA_cross.sh"

Lineage-specific piRNA FASTA files were mapped as in RMD1 using Bowtie v1.1.2 while allowing for no mismatches in the sense orientation and three mismatches in the antisense orientation using the script, "sRNA_map_bowtie.sh."

Count files were then generated fractionally as in RMD1 using the script, "sRNA_Count_Matrix.R" and run using the script, "run_sRNA_Count_Matrix.sh." The lineage-sorted count matrix is imported below.

Load Transcriptome Annotation Matrix and Lineage-sorted piRNA Mapping Results

```
Read_Counts_Master_DF <- read.table("objects/Annotated_piRNA_Degradome_DGE_Count_Matrix.txt",
    sep = "\t")

# Retain only transcript Length and Classification

Length_and_Class <- Read_Counts_Master_DF[, c(1, 3, 11)]

# Import Lineage-specific piRNA count matrix

Bowtie_small_RNAs <- read.table("objects/small_RNA_Count_Matrix.txt", header = T)

Bowtie_small_RNAs <- merge(Length_and_Class, Bowtie_small_RNAs, by = "ID", all = T)
```

Generate Normalized piRNA Mapping Density Values

PIWI targets should have a high density of piRNA counts. We normalize piRNA counts by transcript length to determine piRNA count density.

To determine if the piRNA count density values were significantly different between classes of transcripts, we performed Tukey's Honest Significant Difference test to compare mean piRNA count density between each

transcript type (i.e. TE, ncRNA, Unchar., Gene) for each piRNA class (i.e. Hywi Antisense-mapped, Hyli Sense-mapped, etc.).

```
# Generate RPK values

norm <- (Bowtie_small_RNAs$Length/1000)

Bowtie_small_RNAs[, c(16:27)] <- Bowtie_small_RNAs[, c(4:15)]/norm

colnames(Bowtie_small_RNAs)[16:27] <- c("Endo_Hyli_AS_RPK", "Endo_Hyli_S_RPK", "Endo_Hywi_AS_RPK",
    "Endo_Hywi_S_RPK", "Ecto_Hyli_AS_RPK", "Ecto_Hyli_S_RPK", "Ecto_Hywi_AS_RPK",
    "Ecto_Hywi_S_RPK", "Int_Hyli_AS_RPK", "Int_Hyli_S_RPK", "Int_Hywi_AS_RPK", "Int_Hywi_S_RPK")


# only keep normalized counts
Normalized_Mapping_Counts_Matrix <- Bowtie_small_RNAs[, c(3, 16:27)]

Normalized_Mapping_Counts_Matrix_Formatted <- melt(Normalized_Mapping_Counts_Matrix,
    id.var = "Transcript_Class")

# Subset count density based on piRNA origin
Endo_Hyli_AS_Stats <- subset(Normalized_Mapping_Counts_Matrix_Formatted, variable ==
    "Endo_Hyli_AS_RPK")
Ecto_Hyli_AS_Stats <- subset(Normalized_Mapping_Counts_Matrix_Formatted, variable ==
    "Ecto_Hyli_AS_RPK")
Int_Hyli_AS_Stats <- subset(Normalized_Mapping_Counts_Matrix_Formatted, variable ==
    "Int_Hyli_AS_RPK")

Endo_Hywi_AS_Stats <- subset(Normalized_Mapping_Counts_Matrix_Formatted, variable ==
    "Endo_Hywi_AS_RPK")
Ecto_Hywi_AS_Stats <- subset(Normalized_Mapping_Counts_Matrix_Formatted, variable ==
    "Ecto_Hywi_AS_RPK")
Int_Hywi_AS_Stats <- subset(Normalized_Mapping_Counts_Matrix_Formatted, variable ==
    "Int_Hywi_AS_RPK")

Endo_Hyli_S_Stats <- subset(Normalized_Mapping_Counts_Matrix_Formatted, variable ==
    "Endo_Hyli_S_RPK")
Ecto_Hyli_S_Stats <- subset(Normalized_Mapping_Counts_Matrix_Formatted, variable ==
    "Ecto_Hyli_S_RPK")
Int_Hyli_S_Stats <- subset(Normalized_Mapping_Counts_Matrix_Formatted, variable ==
    "Int_Hyli_S_RPK")

Endo_Hywi_S_Stats <- subset(Normalized_Mapping_Counts_Matrix_Formatted, variable ==
    "Endo_Hywi_S_RPK")
Ecto_Hywi_S_Stats <- subset(Normalized_Mapping_Counts_Matrix_Formatted, variable ==
    "Ecto_Hywi_S_RPK")
Int_Hywi_S_Stats <- subset(Normalized_Mapping_Counts_Matrix_Formatted, variable ==
    "Int_Hywi_S_RPK")


# Develop Tukey Test Function (ANOVA post hoc test)

Tukey_Test <- function(x) {
    res.aov <- aov(value ~ Transcript_Class, data = x)
```

```
    return(TukeyHSD(res.aov))
}

# Run Tukey Test

Tukey_Test(Int_Hywi_AS_Stats)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = value ~ Transcript_Class, data = x)
##
## $Transcript_Class
##                    diff        lwr       upr     p adj
## ncRNA-Gene    11.18870  -1.732110  24.10951 0.1165600
## TE-Gene       87.45788  68.944129 105.97164 0.0000000
## Unchar-Gene   21.68586   9.017549  34.35416 0.0000646
## TE-ncRNA      76.26919  56.088805  96.44956 0.0000000
## Unchar-ncRNA  10.49716  -4.501997  25.49631 0.2742610
## Unchar-TE    -65.77203 -85.791679 -45.75238 0.0000000
```

```
Tukey_Test(Int_Hyli_AS_Stats)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = value ~ Transcript_Class, data = x)
##
## $Transcript_Class
##                     diff        lwr       upr     p adj
## ncRNA-Gene     6.238687  -3.851444  16.32882 0.3852025
## TE-Gene       49.952450  35.494669  64.41023 0.0000000
## Unchar-Gene   16.143014   6.250066  26.03596 0.0001622
## TE-ncRNA      43.713763  27.954479  59.47305 0.0000000
## Unchar-ncRNA   9.904327  -1.808828  21.61748 0.1309726
## Unchar-TE    -33.809435 -49.443203 -18.17567 0.0000002
```

```
Tukey_Test(Int_Hywi_S_Stats)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = value ~ Transcript_Class, data = x)
##
## $Transcript_Class
##                     diff        lwr       upr     p adj
## ncRNA-Gene    15.374030 -11.224967 41.973026 0.4466386
## TE-Gene       47.751361   9.638630 85.864092 0.0070516
## Unchar-Gene    0.103119 -25.976075 26.182313 0.9999996
## TE-ncRNA      32.377331  -9.166343 73.921005 0.1870928
## Unchar-ncRNA -15.270911 -46.148425 15.606604 0.5817569
## Unchar-TE    -47.648242 -88.861038 -6.435445 0.0157555
```

```
Tukey_Test(Int_Hyli_S_Stats)
```

```
##   Tukey multiple comparisons of means
```

```
##     95% family-wise confidence level
##
## Fit: aov(formula = value ~ Transcript_Class, data = x)
##
## $Transcript_Class
##                     diff       lwr        upr     p adj
## ncRNA-Gene    14.661322 -15.08134 44.4039833 0.5844307
## TE-Gene       47.563943   4.94677 90.1811168 0.0215542
## Unchar-Gene    1.677485 -27.48394 30.8389089 0.9988502
## TE-ncRNA      32.902621 -13.55099 79.3562315 0.2639789
## Unchar-ncRNA -12.983838 -47.51068 21.5430087 0.7687600
## Unchar-TE    -45.886459 -91.97009  0.1971687 0.0514769
```

```
Tukey_Test(Endo_Hywi_AS_Stats)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = value ~ Transcript_Class, data = x)
##
## $Transcript_Class
##                    diff         lwr       upr     p adj
## ncRNA-Gene    1.021494  -1.3045665  3.347554 0.6720711
## TE-Gene       9.235563   5.9026365 12.568490 0.0000000
## Unchar-Gene   2.591703   0.3110993  4.872307 0.0184120
## TE-ncRNA      8.214070   4.5811096 11.847030 0.0000000
## Unchar-ncRNA  1.570209  -1.1300038  4.270423 0.4411350
## Unchar-TE    -6.643860 -10.2478852 -3.039835 0.0000130
```

```
Tukey_Test(Endo_Hyli_AS_Stats)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = value ~ Transcript_Class, data = x)
##
## $Transcript_Class
##                    diff        lwr      upr     p adj
## ncRNA-Gene   -1.206593 -26.212714 23.79953 0.9993198
## TE-Gene       4.318266 -31.512092 40.14862 0.9897142
## Unchar-Gene  16.216851  -8.300596 40.73430 0.3240012
## TE-ncRNA      5.524859 -33.530981 44.58070 0.9835894
## Unchar-ncRNA 17.423444 -11.604978 46.45187 0.4123082
## Unchar-TE    11.898585 -26.846192 50.64336 0.8595151
```

```
Tukey_Test(Endo_Hywi_S_Stats)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = value ~ Transcript_Class, data = x)
##
## $Transcript_Class
##                     diff       lwr       upr     p adj
## ncRNA-Gene   -17.233378 -47.04750 12.580743 0.4465845
## TE-Gene       -8.965788 -51.68535 33.753778 0.9494516
```

```
## Unchar-Gene  -20.776673 -50.00816  8.454815 0.2610110
## TE-ncRNA       8.267590 -38.29763 54.832810 0.9684524
## Unchar-ncRNA  -3.543295 -38.15310 31.066506 0.9936320
## Unchar-TE    -11.810885 -58.00523 34.383464 0.9131739
```

```
Tukey_Test(Endo_Hyli_S_Stats)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = value ~ Transcript_Class, data = x)
##
## $Transcript_Class
##                    diff       lwr      upr     p adj
## ncRNA-Gene   -18.216985 -53.09706 16.66309 0.5362233
## TE-Gene      -12.067538 -62.04592 37.91085 0.9256247
## Unchar-Gene  -23.565493 -57.76394 10.63295 0.2877255
## TE-ncRNA       6.149447 -48.32804 60.62693 0.9915117
## Unchar-ncRNA  -5.348508 -45.83913 35.14212 0.9865544
## Unchar-TE    -11.497955 -65.54155 42.54564 0.9474980
```

```
Tukey_Test(Ecto_Hywi_AS_Stats)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = value ~ Transcript_Class, data = x)
##
## $Transcript_Class
##                    diff        lwr      upr     p adj
## ncRNA-Gene    0.8620878 -1.6124419 3.336617 0.8074515
## TE-Gene       4.0776170  0.5319537 7.623280 0.0165405
## Unchar-Gene   3.3584078  0.9322358 5.784580 0.0021309
## TE-ncRNA      3.2155292 -0.6493179 7.080376 0.1413038
## Unchar-ncRNA  2.4963200 -0.3762443 5.368884 0.1144932
## Unchar-TE    -0.7192092 -4.5532745 3.114856 0.9631250
```

```
Tukey_Test(Ecto_Hyli_AS_Stats)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = value ~ Transcript_Class, data = x)
##
## $Transcript_Class
##                     diff       lwr      upr     p adj
## ncRNA-Gene   -0.93451025 -4.648886 2.779865 0.9168589
## TE-Gene       1.77916290 -3.543030 7.101356 0.8260438
## Unchar-Gene   1.83076096 -1.811027 5.472549 0.5683538
## TE-ncRNA      2.71367315 -3.087629 8.514975 0.6257378
## Unchar-ncRNA  2.76527121 -1.546571 7.077114 0.3518877
## Unchar-TE     0.05159805 -5.703499 5.806695 0.9999956
```

```
Tukey_Test(Ecto_Hywi_S_Stats)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
```

```
##
## Fit: aov(formula = value ~ Transcript_Class, data = x)
##
## $Transcript_Class
##                    diff       lwr       upr     p adj
## ncRNA-Gene   -5.189888 -20.15224  9.772470 0.8094971
## TE-Gene      -1.341487 -22.78050 20.097529 0.9985224
## Unchar-Gene  -7.395514 -22.06547  7.274446 0.5660225
## TE-ncRNA      3.848401 -19.52058 27.217376 0.9745554
## Unchar-ncRNA -2.205627 -19.57472 15.163465 0.9880155
## Unchar-TE    -6.054027 -29.23688 17.128825 0.9081087
```

```
Tukey_Test(Ecto_Hyli_S_Stats)
```

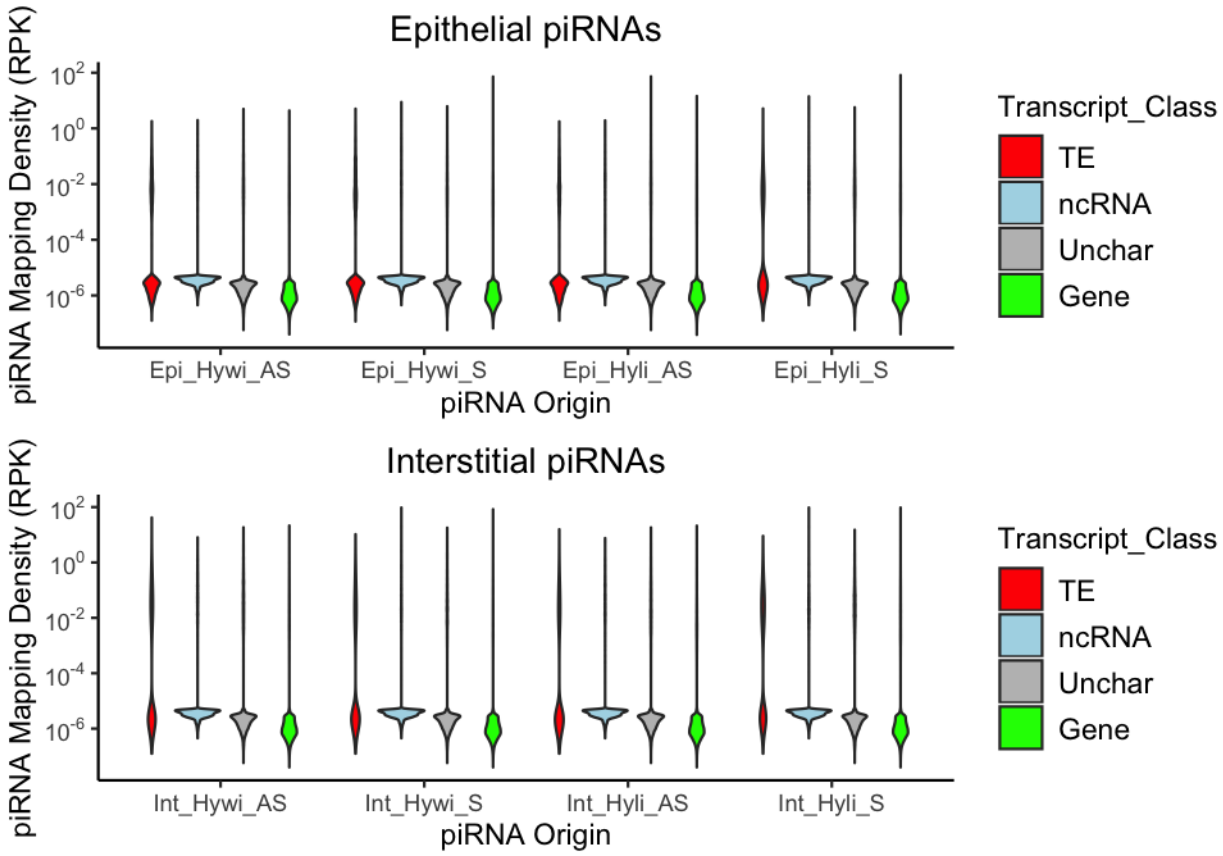
```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = value ~ Transcript_Class, data = x)
##
## $Transcript_Class
##                     diff       lwr       upr     p adj
## ncRNA-Gene   -4.8736875 -24.31336 14.565989 0.9176453
## TE-Gene      -4.4537088 -32.30811 23.400695 0.9766221
## Unchar-Gene  -9.7609653 -28.82075  9.298818 0.5528750
## TE-ncRNA      0.4199787 -29.94190 30.781861 0.9999839
## Unchar-ncRNA -4.8872778 -27.45388 17.679323 0.9448392
## Unchar-TE    -5.3072565 -35.42732 24.812807 0.9691284
```

Visualizing piRNA Mapping

Since the range of observed count density values was large, we used a log scale to visualize piRNA count density. For violin plot visualization, we added a pseudocount to the raw piRNA counts to remove any 0 count density values that would return infinite values on a log scale. Violin plots allow for the visualization of high mapping outliers in the TE category. The pseudocount we chose was 0.001 since that approximated the lowest fractional count administered by our counting strategy. We explored piRNA count density for 1) Interstitial piRNAs and 2) Epithelial piRNAs.

```
# Create pseudocount
pseudocount <- 0.001

# Create a count matrix with transcript length and classification retained
count_matrix <- Bowtie_small_RNAs[, c(4:15, 2, 3)]

# Take the average of Endo and Ecto to generate Epithelial Counts
count_matrix$Epi_Hyli_AS <- (count_matrix$Endo_Hyli_AS + count_matrix$Ecto_Hyli_AS)/2
count_matrix$Epi_Hyli_S <- (count_matrix$Endo_Hyli_S + count_matrix$Ecto_Hyli_S)/2

count_matrix$Epi_Hywi_AS <- (count_matrix$Endo_Hywi_AS + count_matrix$Ecto_Hywi_AS)/2
count_matrix$Epi_Hywi_S <- (count_matrix$Endo_Hywi_S + count_matrix$Ecto_Hywi_S)/2


# Add pseudocount to piRNA counts then generate piRNA count density values (RPK)
count_matrix[, c(9:12, 15:18)] <- count_matrix[, c(9:12, 15:18)] + pseudocount
count_matrix[, c(9:12, 15:18)] <- count_matrix[, c(9:12, 15:18)]/count_matrix$Length
```

```r
# Plot intersititial piRNA count density values
Int_matrix_data <- count_matrix[, c(9:12, 14)]
Int_matrix_data_formatted <- melt(Int_matrix_data, id.var = "Transcript_Class")
colnames(Int_matrix_data_formatted) <- c("Transcript_Class", "piRNA_Origin", "piRNA_Mapping_Density")
Int_matrix_data_formatted$Transcript_Class <- factor(Int_matrix_data_formatted$Transcript_Class,
    levels = c("TE", "ncRNA", "Unchar", "Gene"))
Int_level_order <- c("Int_Hywi_AS", "Int_Hywi_S", "Int_Hyli_AS", "Int_Hyli_S")

Int_violin <- ggplot(data = Int_matrix_data_formatted, aes(x = factor(piRNA_Origin,
    level = Int_level_order), y = piRNA_Mapping_Density), log = "y") + geom_violin(aes(fill = Transcrip
    scale_y_log10(breaks = scales::trans_breaks("log10", function(x) 10^x), labels = scales::trans_forma
        scales::math_format(10^.x)))
Int_plot <- Int_violin + scale_fill_manual(values = c("red", "light blue", "grey",
    "green")) + theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
    panel.background = element_blank(), axis.line = element_line(colour = "black")) +
    theme(legend.text = element_text(size = rel(1))) + ggtitle("Interstitial piRNAs") +
    theme(plot.title = element_text(hjust = 0.5)) + xlab("piRNA Origin") + ylab("piRNA Mapping Density

# Plot epithelial piRNA count density values

Epi_matrix_data <- count_matrix[, c(15:18, 14)]
Epi_matrix_data_formatted <- melt(Epi_matrix_data, id.var = "Transcript_Class")
colnames(Epi_matrix_data_formatted) <- c("Transcript_Class", "piRNA_Origin", "piRNA_Mapping_Density")
Epi_matrix_data_formatted$Transcript_Class <- factor(Epi_matrix_data_formatted$Transcript_Class,
    levels = c("TE", "ncRNA", "Unchar", "Gene"))
Epi_level_order <- c("Epi_Hywi_AS", "Epi_Hywi_S", "Epi_Hyli_AS", "Epi_Hyli_S")


Epi_violin <- ggplot(data = Epi_matrix_data_formatted, aes(x = factor(piRNA_Origin,
    level = Epi_level_order), y = piRNA_Mapping_Density), log = "y") + geom_violin(aes(fill = Transcrip
    scale_y_log10(breaks = scales::trans_breaks("log10", function(x) 10^x), labels = scales::trans_forma
        scales::math_format(10^.x)))
Epi_plot <- Epi_violin + scale_fill_manual(values = c("red", "light blue", "grey",
    "green")) + theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
    panel.background = element_blank(), axis.line = element_line(colour = "black")) +
    theme(legend.text = element_text(size = rel(1))) + ggtitle("Epithelial piRNAs") +
    theme(plot.title = element_text(hjust = 0.5)) + xlab("piRNA Origin") + ylab("piRNA Mapping Density

# Arrange
ggarrange(Epi_plot, Int_plot, ncol = 1, nrow = 2)
```



Explore Lineage-sorted piRNA Diversity

To explore the diversity of piRNAs in different lineages, we identified unique piRNA sequences in each lineage as well as the piRNAs species that were present in multiple lineages.

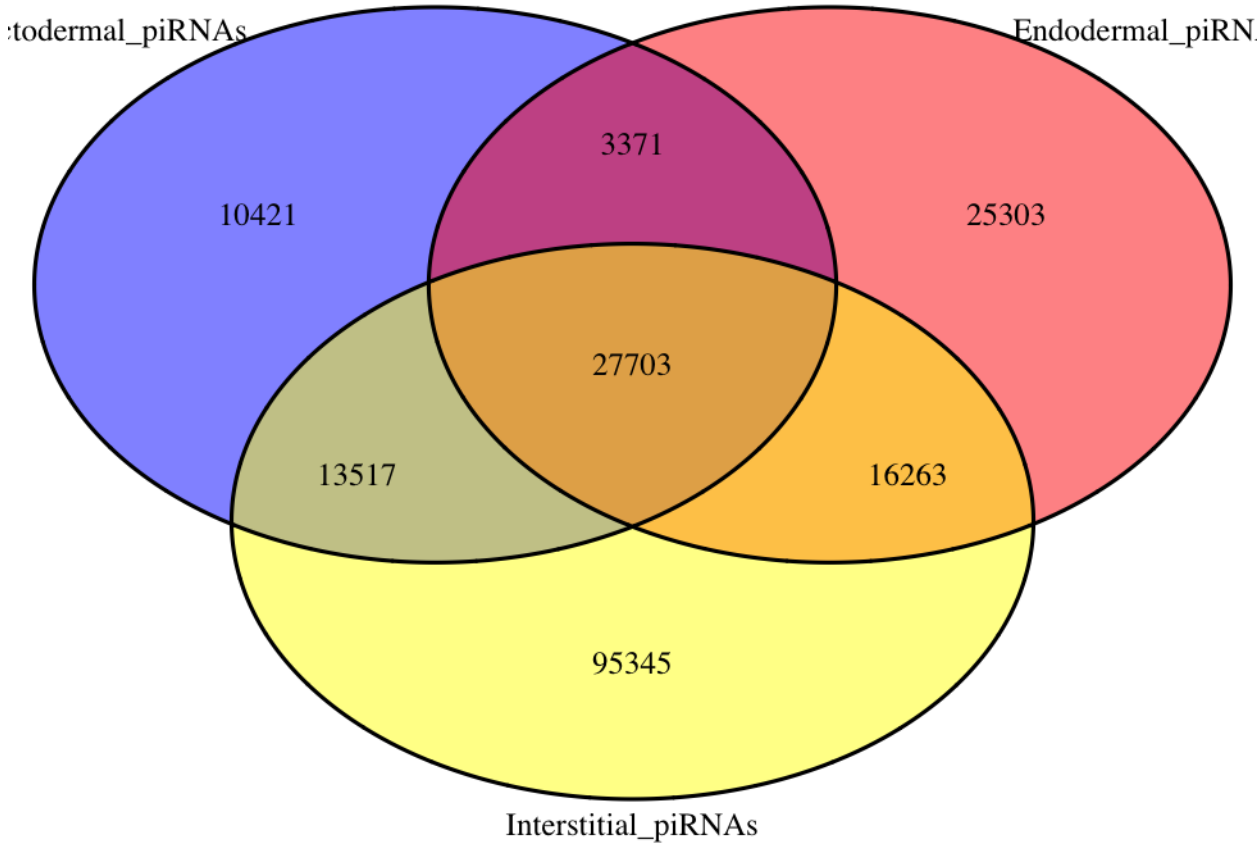
Lineage-sorted unique sequence lists were generated from the lineage-sorted piRNA libraries generated previously. Each unique sequence was collapsed down to one representative sequence using the script, "unique_sRNA_piRNA_gen_separate.R" and run using the script, "run_unique_sRNA_piRNA_gen_separate.sh."

Lineage-specific and shared Hywi and Hyli piRNAs were visualized using venn diagrams.

Unique Hywi piRNA Sequences

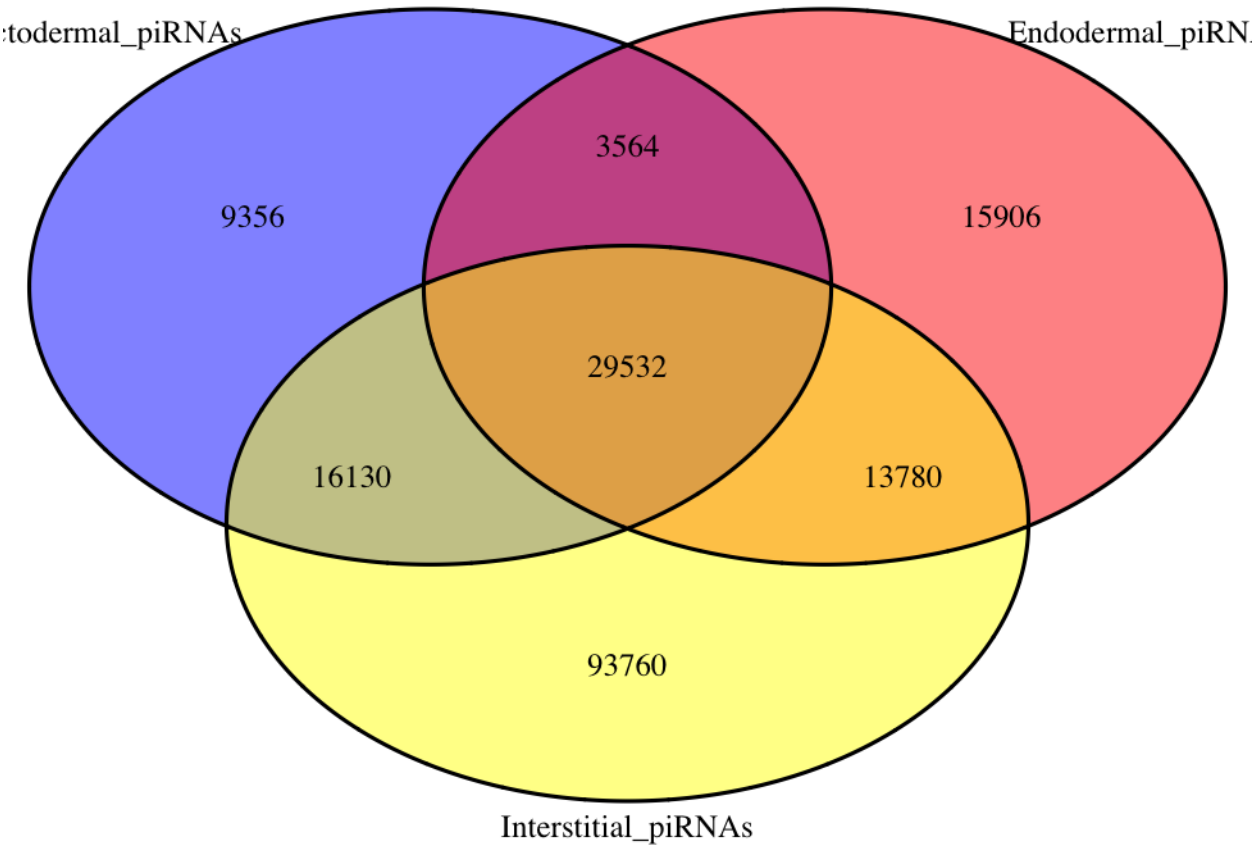
```
# Load unique piRNA sequences sorted by lineage and protein origin

load("objects/Unduplicated_Ecto_Hywi_small_RNAs_crossref.Rda")
load("objects/Unduplicated_Endo_Hywi_small_RNAs_crossref.Rda")
load("objects/Unduplicated_Int_Hywi_small_RNAs_crossref.Rda")

# Visualize shared and unique piRNA species by lineage

Venn_Hywi <- list(Ectodermal_piRNAs = Ecto_Hywi$seq, Endodermal_piRNAs = Endo_Hywi$seq,
    Interstitial_piRNAs = Int_Hywi$seq)
```

```
venn.plot.hywi <- venn.diagram(Venn_Hywi, filename = NULL, fill = c("blue", "red",
    "yellow"))
grid.draw(venn.plot.hywi)
```



Unique Hyli piRNA Sequences

```
load("objects/Unduplicated_Ecto_Hyli_small_RNAs_crossref.Rda")
load("objects/Unduplicated_Endo_Hyli_small_RNAs_crossref.Rda")
load("objects/Unduplicated_Int_Hyli_small_RNAs_crossref.Rda")

# Visualize shared and unique piRNA species by lineage

Venn_Hyli <- list(Ectodermal_piRNAs = Ecto_Hyli$seq, Endodermal_piRNAs = Endo_Hyli$seq,
    Interstitial_piRNAs = Int_Hyli$seq)

venn.plot.hyli <- venn.diagram(Venn_Hyli, filename = NULL, fill = c("blue", "red",
    "yellow"))
grid.draw(venn.plot.hyli)
```



Software versions

This document was computed on Fri Jan 10 10:47:23 2020 with the following R package versions.

```
R version 3.5.3 (2019-03-11)
Platform: x86_64-apple-darwin15.6.0 (64-bit)
Running under: macOS Mojave 10.14.5

Matrix products: default
BLAS: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/3.5/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] grid      stats     graphics  grDevices utils     datasets  methods
[8] base

other attached packages:
[1] VennDiagram_1.6.20  futile.logger_1.4.3 ggpubr_0.2.4
[4] magrittr_1.5        ggplot2_3.2.0       reshape2_1.4.3
[7] dplyr_0.8.3         knitr_1.22
```

```
loaded via a namespace (and not attached):
 [1] Rcpp_1.0.1           pillar_1.4.2         compiler_3.5.3
 [4] formatR_1.7          plyr_1.8.4           futile.options_1.0.1
 [7] tools_3.5.3          digest_0.6.20        evaluate_0.13
[10] tibble_2.1.3         gtable_0.3.0         pkgconfig_2.0.2
[13] rlang_0.4.0          yaml_2.2.0           xfun_0.5
[16] withr_2.1.2          stringr_1.4.0        tidyselect_0.2.5
[19] cowplot_0.9.4        glue_1.3.1           R6_2.4.0
[22] rmarkdown_1.12       purrr_0.3.2          lambda.r_1.2.3
[25] scales_1.0.0         htmltools_0.3.6      assertthat_0.2.1
[28] colorspace_1.4-1     ggsignif_0.5.0       labeling_0.3
[31] stringi_1.4.3        lazyeval_0.2.2       munsell_0.5.0
[34] crayon_1.3.4
```

5 Single Cell Data Exploration

Bryan Teefy

12/20/2019

Load required libraries

```
library(URD)
library(Seurat)
```

Single Cell Data Exploration

We analyzed the homeostatic expression patterns of the 24 high mapping gene transcripts upregulated in response to hywi knockdown by interrogating single-cell expression data (Siebert, 2019). To interrogate expression, we used URD spline objects for ectoderm and endoderm. These objects contain expression data for genes that are expressed in at least 1% of the ectodermal or endodermal epithelial cells. We found epithelial expression for 16 of the 24 putative gene targets: 1) 13 gene transcripts are expressed in both the endodermal and ectodermal epithelial lineages, 2) one transcript is expressed only in the ectodermal epithelial lineage, and 3) two transcripts are expressed only in the endodermal epithelial lineage. No epithelial expression was found for 8 of the gene transcripts, which could indicate low expression in a homeostatic animal. Higher expression at the extremities in both the ectoderm and endoderm and lower expression in body regions would be consistent with Hywi-mediated repression. Putative targets could show high expression. The gene, t14391, shows such a pattern.

Load the Necessary Files

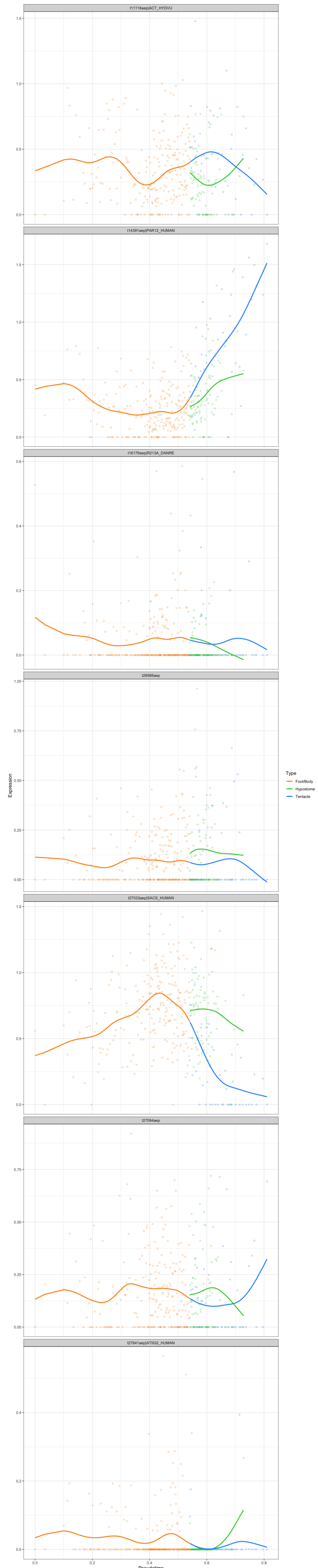
```
# Load the list of searchable high mapping genes for each lineage

endo_mappers <- read.table("objects/Endodermal_High_Mappers.txt", header = F)
colnames(endo_mappers) <- "ID"

ecto_mappers <- read.table("objects/Ectodermal_High_Mappers.txt", header = F)
colnames(ecto_mappers) <- "ID"

# Load the seurat object to load the searchable transcript IDs The seurat object
# (file Hydra_Seurat_Whole_Transcriptome) can be downloaded from Dryad:
# https://doi.org/10.5061/dryad.v5r6077. After download, place in the 'objects'
# folder

Hydra_Object <- readRDS("objects/Hydra_Seurat_Whole_Transcriptome.rds")

# Load single cell pseudotime objects (spline objects) and High Mapping Gene List
# The spline objects (file Hydra_URD_analysis_objects) can be downloaded from
# Dryad: https://doi.org/10.5061/dryad.v5r6077. After download, place in the
# 'objects' folder
```

```
endoderm.splines <- readRDS("objects/Splines-Endoderm.rds")
ectoderm.splines <- readRDS("objects/Splines-Ectoderm.rds")
```

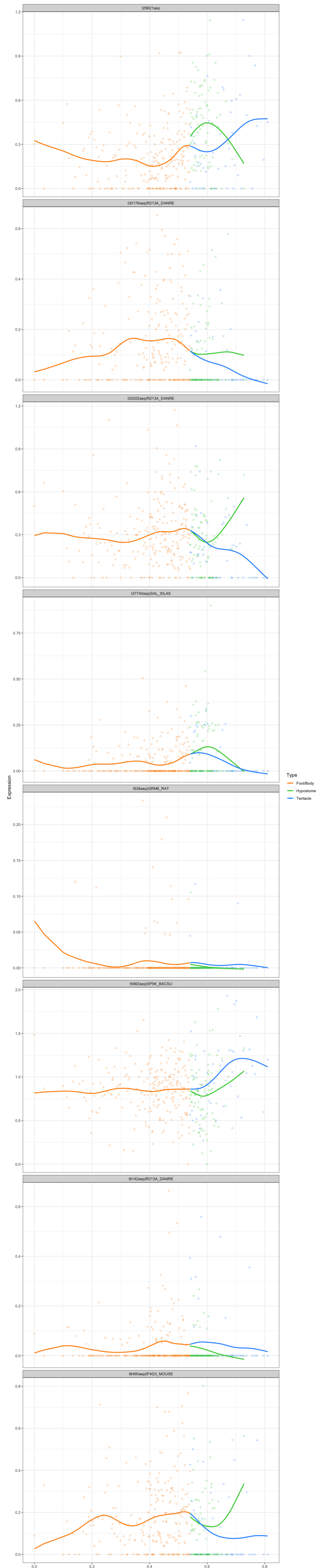
Data Exploration

```
# Establish hFind function to return the searchable transcript IDs

hFind <- function(x) {
    return(Hydra_Object@data@Dimnames[[1]][grep(x, Hydra_Object@data@Dimnames[[1]],
        ignore.case = T)])
}

# Run hFind function to convert to searchable transcript IDs

endo_mappers$ID <- sapply(endo_mappers$ID, FUN = hFind)

ecto_mappers$ID <- sapply(ecto_mappers$ID, FUN = hFind)
```

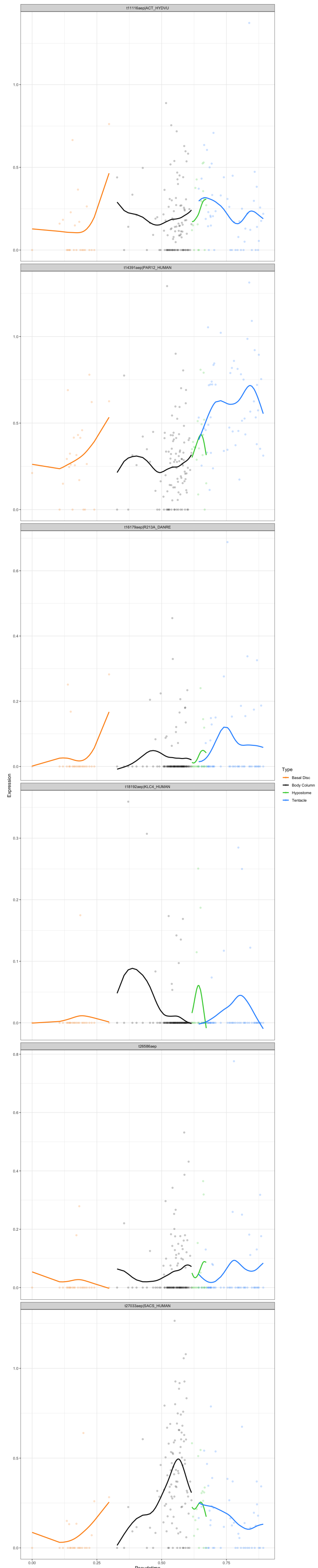
```
# Generate Endodermal and Endodermal Plots

plotSmoothFitMultiCascade(endoderm.splines, genes = endo_mappers$ID[1:7], scaled = F,
    colors = c(`Foot/Body` = "#FF8C00", Tentacle = "#1E90FF", Hypostome = "#32CD32"),
    ncol = 1)
```


t11116aep|ACT_HYDVU
t14391aep|PAR12_HUMAN
t16179aep|R213A_DANRE
t26586aep
t27033aep|SACS_HUMAN
t27094aep
t27641aep|AT5G2_HUMAN
Expression
Pseudotime
Type
Foot/Body
Hypostome
Tentacle

```r
plotSmoothFitMultiCascade(endoderm.splines, genes = endo_mappers$ID[8:15], scaled = F,
    colors = c(`Foot/Body` = "#FF8C00", Tentacle = "#1E90FF", Hypostome = "#32CD32"),
    ncol = 1)
```

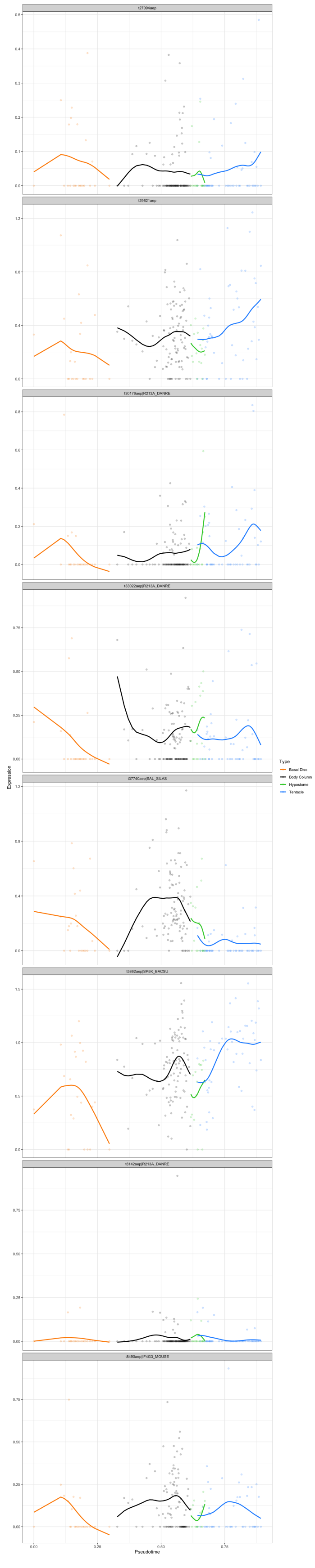

t29621aep
t30176aep|R213A_DANRE
t33022aep|R213A_DANRE
t37740aep|SAL_SILAS
t536aep|GRM8_RAT
t5862aep|SP5K_BACSU
t8142aep|R213A_DANRE
t8490aep|IF4G3_MOUSE
Expression
Pseudotime
Type
Foot/Body
Hypostome
Tentacle

```
plotSmoothFitMultiCascade(ectoderm.splines, genes = ecto_mappers$ID[1:6], scaled = F,
    colors = c(`Basal Disc` = "#FF8C00", `Body Column` = "#000000", Tentacle = "#1E90FF",
        Hypostome = "#32CD32"), ncol = 1)
```



t11116aep|ACT_HYDVU
t14391aep|PAR12_HUMAN
t16179aep|R213A_DANRE
t18192aep|KLC4_HUMAN
t26586aep
t27033aep|SACS_HUMAN
Expression
Pseudotime
Type
Basal Disc
Body Column
Hypostome
Tentacle

```
plotSmoothFitMultiCascade(ectoderm.splines, genes = ecto_mappers$ID[7:14], scaled = F,
    colors = c(`Basal Disc` = "#FF8C00", `Body Column` = "#000000", Tentacle = "#1E90FF",
        Hypostome = "#32CD32"), ncol = 1)
```



t27094aep
t29621aep
t30176aep|R213A_DANRE
t33022aep|R213A_DANRE
t37740aep|SAL_SILAS
t5862aep|SP5K_BACSU
t8142aep|R213A_DANRE
t8490aep|IF4G3_MOUSE
Expression
Pseudotime
Type
Basal Disc
Body Column
Hypostome
Tentacle

Software versions

This document was computed on Thu Jan 09 15:24:08 2020 with the following R package versions.

```
R version 3.5.3 (2019-03-11)
Platform: x86_64-apple-darwin15.6.0 (64-bit)
Running under: macOS Mojave 10.14.5

Matrix products: default
BLAS: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/3.5/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] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] Seurat_2.3.4  cowplot_0.9.4 URD_1.0.3     Matrix_1.2-16 ggplot2_3.2.0
[6] knitr_1.22

loaded via a namespace (and not attached):
  [1] reticulate_1.11.1           R.utils_2.8.0
  [3] tidyselect_0.2.5            htmlwidgets_1.3
  [5] grid_3.5.3                  trimcluster_0.1-2.1
  [7] ranger_0.11.2               BiocParallel_1.14.2
  [9] Rtsne_0.15                  munsell_0.5.0
 [11] destiny_2.10.2              codetools_0.2-16
 [13] ica_1.0-2                   withr_2.1.2
 [15] colorspace_1.4-1            Biobase_2.40.0
 [17] rstudioapi_0.9.0            stats4_3.5.3
 [19] ROCR_1.0-7                  robustbase_0.93-3
 [21] dtw_1.20-1                  vcd_1.4-4
 [23] VIM_4.8.0                   TTR_0.23-4
 [25] gbRd_0.4-11                 labeling_0.3
 [27] Rdpack_0.10-1               lars_1.2
 [29] GenomeInfoDbData_1.1.0      polyclip_1.10-0
 [31] bit64_0.9-7                 farver_1.1.0
 [33] xfun_0.5                    ggthemes_4.2.0
 [35] diptest_0.75-7              R6_2.4.0
 [37] GenomeInfoDb_1.16.0         RcppEigen_0.3.3.5.0
 [39] hdf5r_1.0.1                 flexmix_2.3-15
 [41] bitops_1.0-6                DelayedArray_0.6.6
 [43] assertthat_0.2.1            SDMTools_1.1-221
 [45] scales_1.0.0                ggraph_1.0.2
 [47] nnet_7.3-12                 gtable_0.3.0
 [49] npsurv_0.4-0                rlang_0.4.0
 [51] scatterplot3d_0.3-41        splines_3.5.3
 [53] lazyeval_0.2.2              acepack_1.4.1
 [55] checkmate_1.9.1             yaml_2.2.0
 [57] reshape2_1.4.3              abind_1.4-5
 [59] backports_1.1.4             Hmisc_4.2-0
```

```
 [61] tools_3.5.3                gplots_3.0.1.1
 [63] RColorBrewer_1.1-2         proxy_0.4-23
 [65] BiocGenerics_0.26.0        ggridges_0.5.1
 [67] Rcpp_1.0.1                 plyr_1.8.4
 [69] base64enc_0.1-3            zlibbioc_1.26.0
 [71] purrr_0.3.2                RCurl_1.95-4.12
 [73] rpart_4.1-13               pbapply_1.4-0
 [75] viridis_0.5.1              S4Vectors_0.18.3
 [77] zoo_1.8-4                  SummarizedExperiment_1.10.1
 [79] haven_2.1.0                ggrepel_0.8.1
 [81] cluster_2.0.7-1            magrittr_1.5
 [83] data.table_1.12.2          openxlsx_4.1.0.1
 [85] gmodels_2.18.1             lmtest_0.9-36
 [87] RANN_2.6.1                 mvtnorm_1.0-10
 [89] fitdistrplus_1.0-14        matrixStats_0.54.0
 [91] hms_0.4.2                  lsei_1.2-0
 [93] evaluate_0.13              smoother_1.1
 [95] rio_0.5.16                 mclust_5.4.2
 [97] readxl_1.3.1               IRanges_2.14.12
 [99] gridExtra_2.3              compiler_3.5.3
[101] tibble_2.1.3               KernSmooth_2.23-15
[103] crayon_1.3.4               R.oo_1.22.0
[105] htmltools_0.3.6            segmented_0.5-3.0
[107] Formula_1.2-3              snow_0.4-3
[109] tidyr_0.8.3                tweenr_1.0.1
[111] formatR_1.7                MASS_7.3-51.1
[113] fpc_2.1-11.1               boot_1.3-20
[115] car_3.0-3                  R.methodsS3_1.7.1
[117] gdata_2.18.0               parallel_3.5.3
[119] metap_1.1                  igraph_1.2.4.1
[121] GenomicRanges_1.32.7       forcats_0.4.0
[123] pkgconfig_2.0.2            foreign_0.8-71
[125] laeken_0.5.0               sp_1.3-1
[127] foreach_1.4.4              XVector_0.20.0
[129] minpack.lm_1.2-1           bibtex_0.4.2
[131] stringr_1.4.0              digest_0.6.20
[133] tsne_0.1-3                 rmarkdown_1.12
[135] cellranger_1.1.0           htmlTable_1.13.1
[137] curl_3.3                   kernlab_0.9-27
[139] gtools_3.8.1               modeltools_0.2-22
[141] jsonlite_1.6               nlme_3.1-137
[143] carData_3.0-2              viridisLite_0.3.0
[145] pillar_1.4.2               lattice_0.20-38
[147] httr_1.4.0                 DEoptimR_1.0-8
[149] survival_2.43-3            glue_1.3.1
[151] xts_0.11-2                 zip_2.0.3
[153] png_0.1-7                  prabclus_2.2-7
[155] iterators_1.0.10           bit_1.1-14
[157] ggforce_0.2.2              class_7.3-15
[159] stringi_1.4.3              mixtools_1.1.0
[161] doSNOW_1.0.16              latticeExtra_0.6-28
[163] caTools_1.17.1.2           dplyr_0.8.3
[165] irlba_2.3.3                e1071_1.7-2
[167] ape_5.2
```