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.

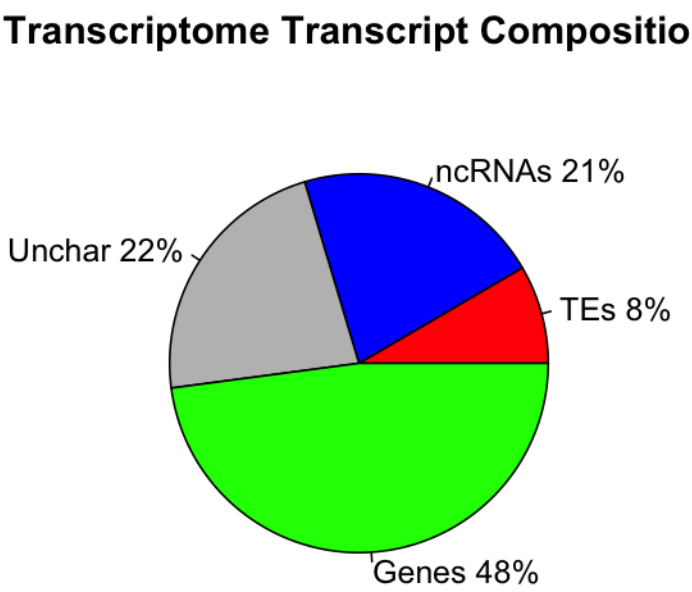
Taxonomically Restricted Genes (TRGs)/Uncharacterized Genes

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)
```

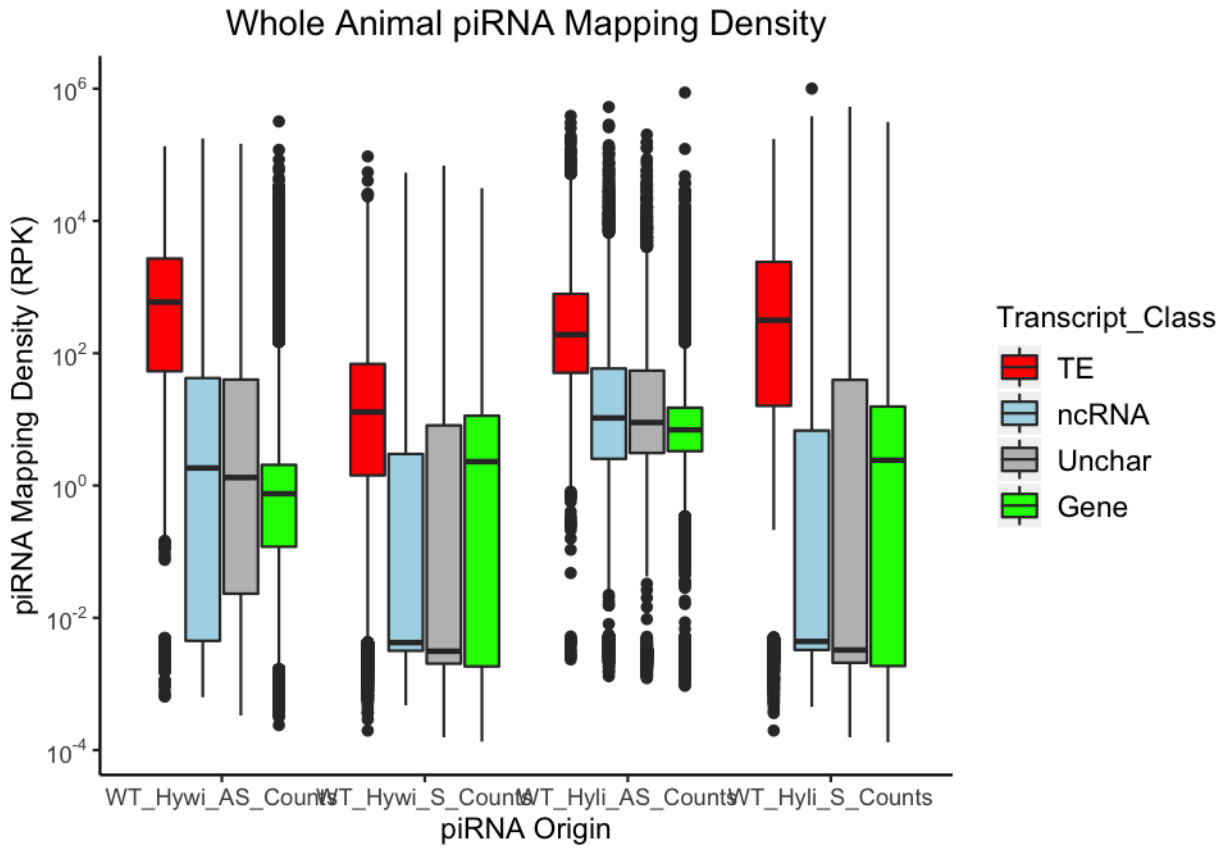
```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = value ~ Transcript_Class, data = x)
##
## $Transcript_Class
##                      diff        lwr         upr     p adj
## ncRNA-Gene    -70.675926 -129.30100  -12.050851 0.0105377
## TE-Gene       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.

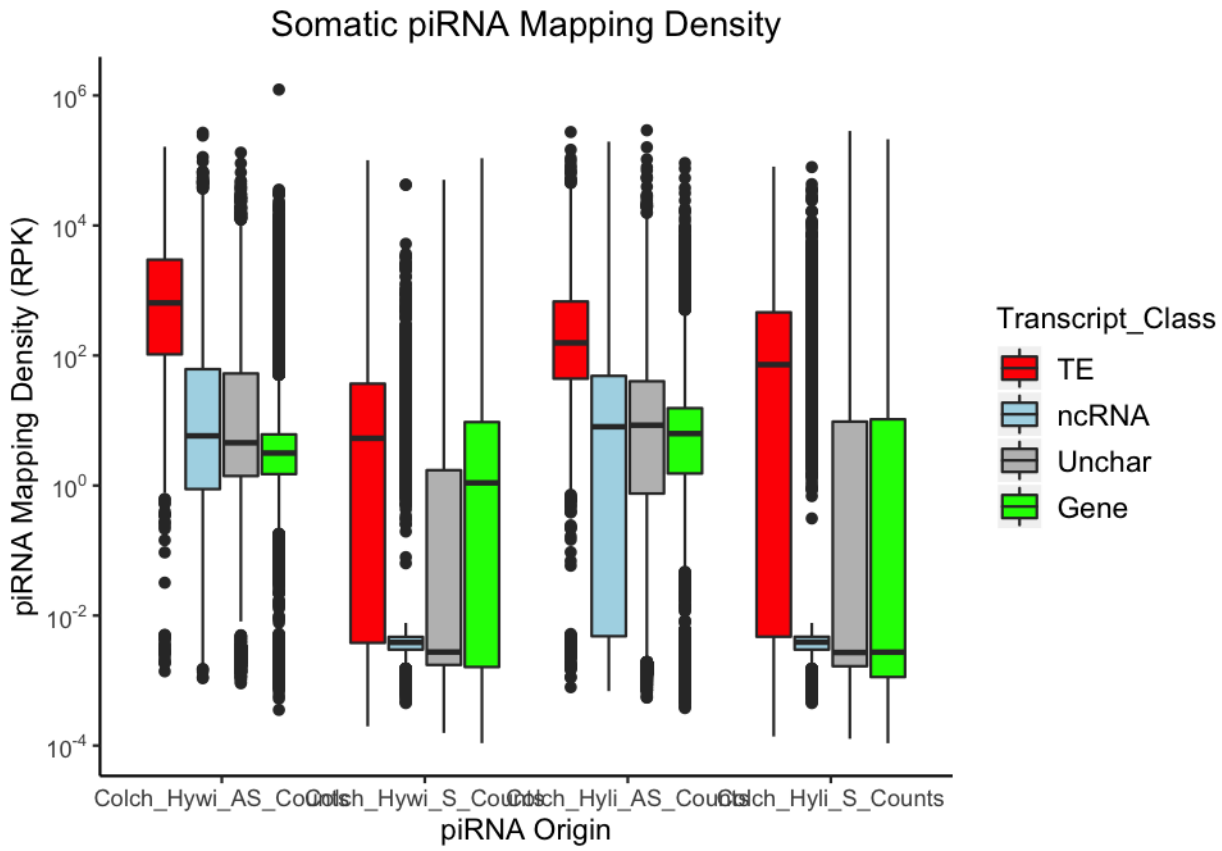
```
# Set pseudocount

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
```