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".

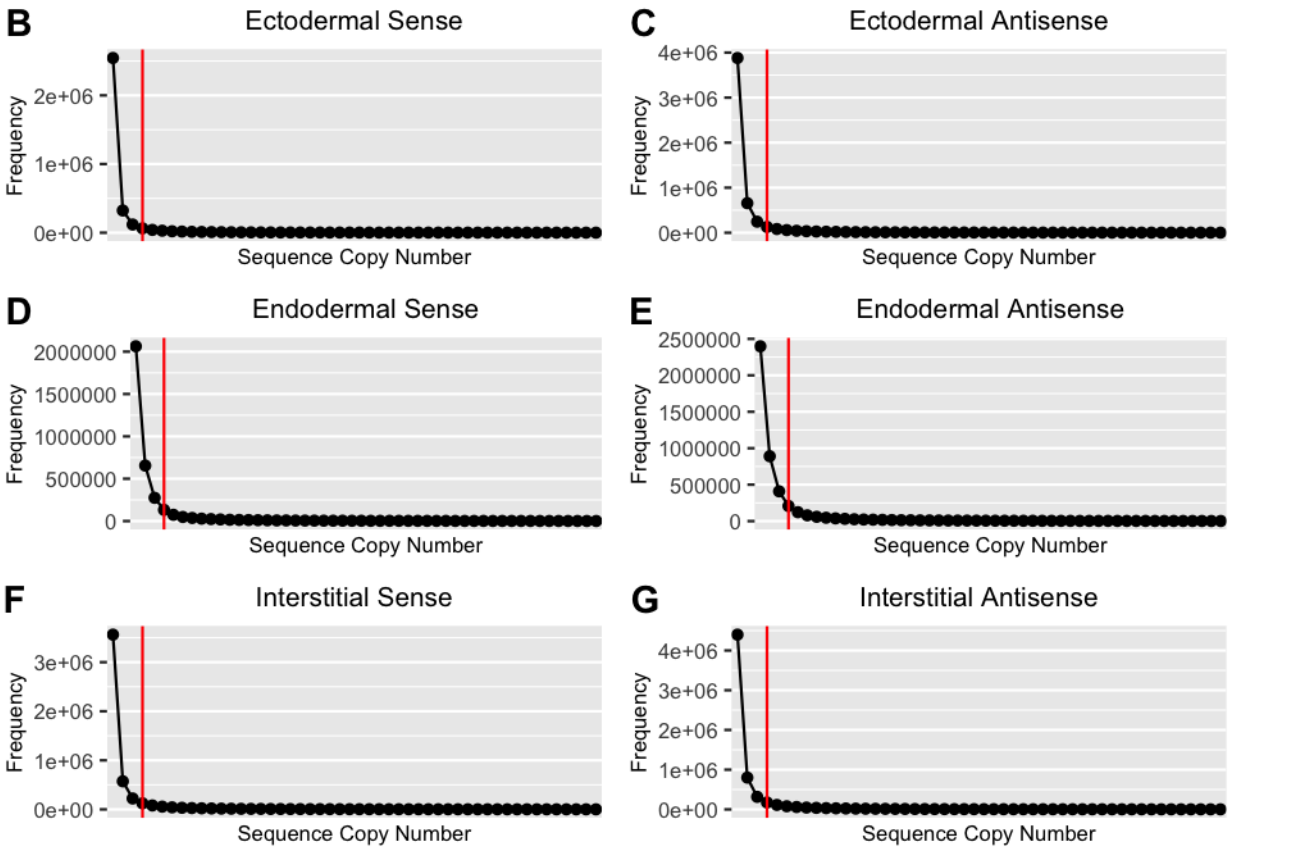
Plots of sequence copy number frequency were generated using the R script, "plot_gen.R" and run with the script, "run_plot_gen.sh."

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)
```

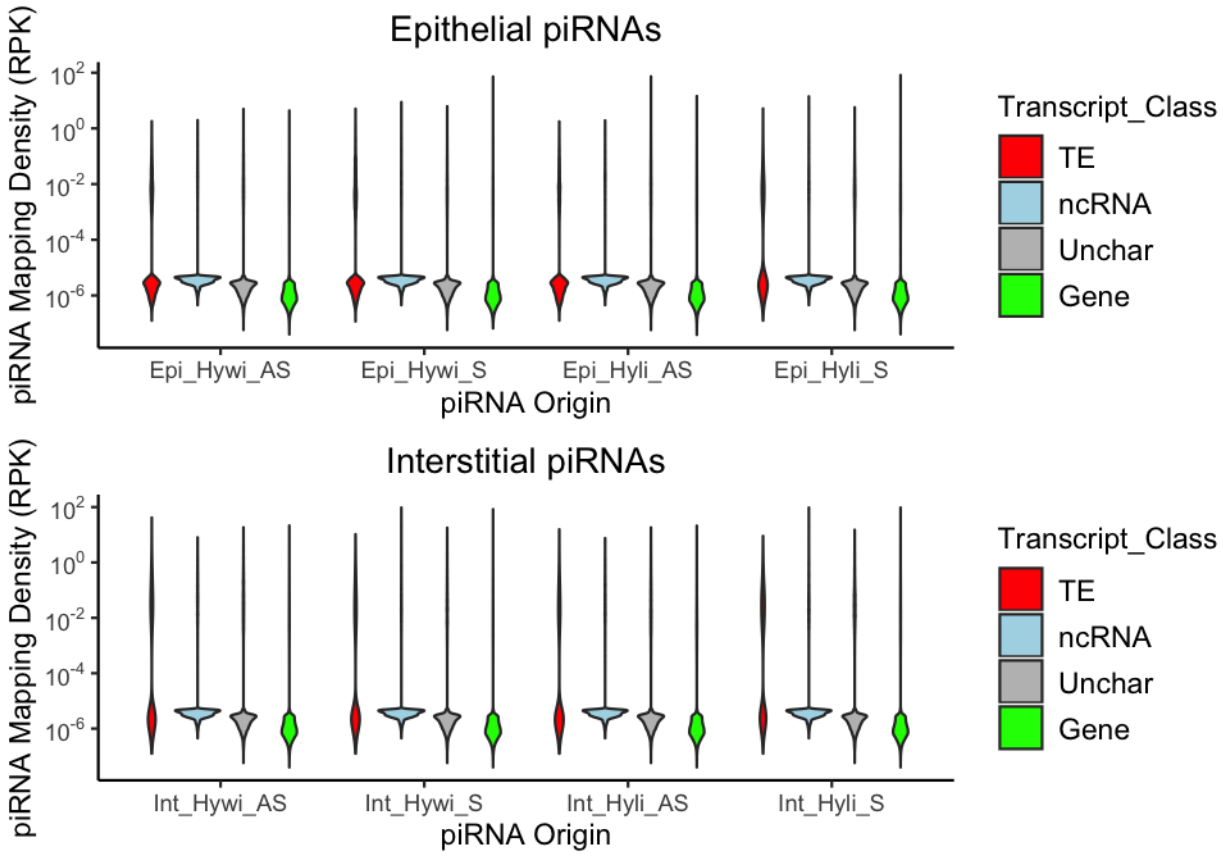
```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = value ~ Transcript_Class, data = x)
##
## $Transcript_Class
##                     diff       lwr       upr     p adj
## ncRNA-Gene   -4.8736875 -24.31336 14.565989 0.9176453
## TE-Gene      -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.

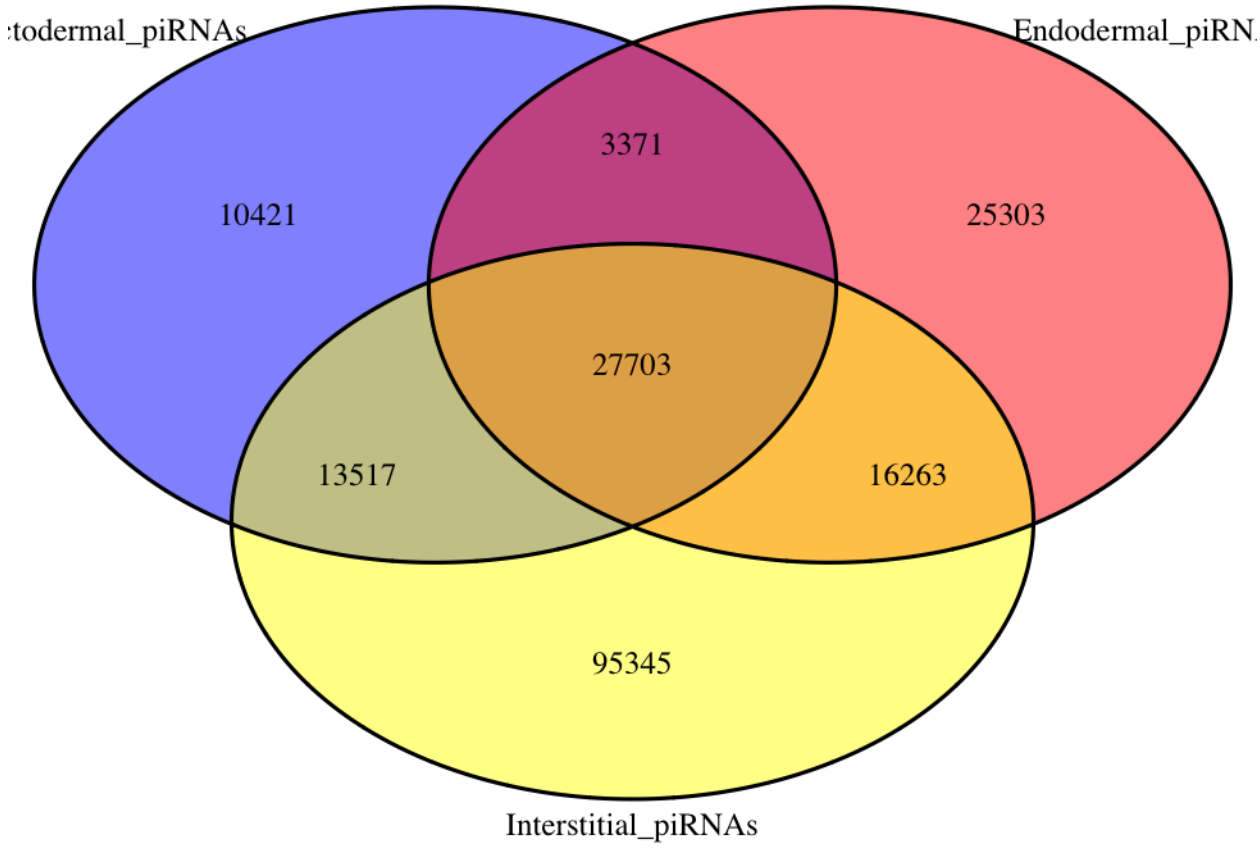
Lineage-sorted unique sequence lists were generated from the lineage-sorted piRNA libraries generated previously. Each unique sequence was collapsed down to one representative sequence using the script, "unique_sRNA_piRNA_gen_separate.R" and run using the script, "run_unique_sRNA_piRNA_gen_separate.sh."

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

Unique Hywi piRNA Sequences

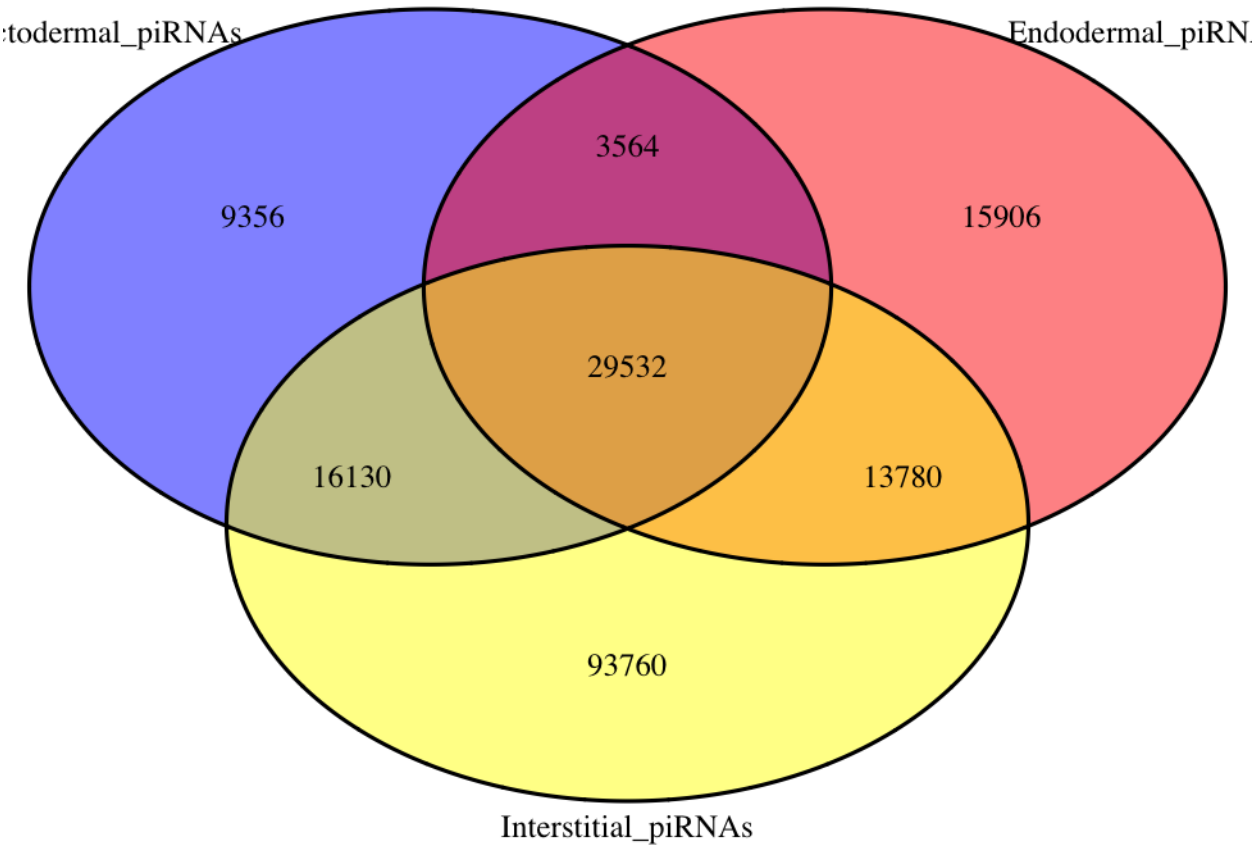
```
# Load unique piRNA sequences sorted by lineage and protein origin

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
```