# 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")
```

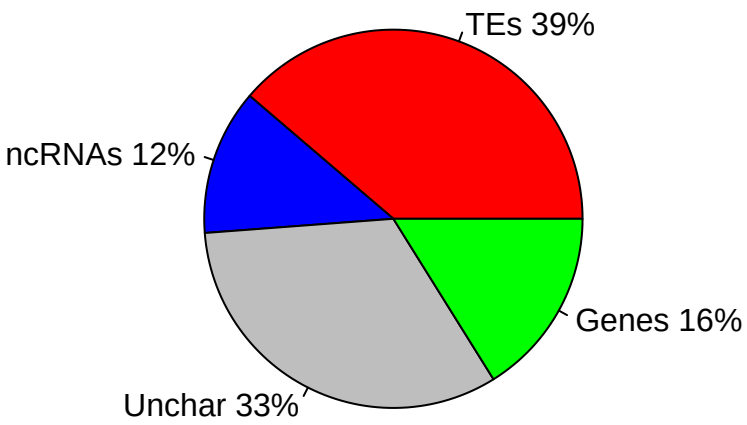
## Load Necessary Packages

```
###load packages###

library(ggplot2)

library(reshape2)

library(ggpubr)
```

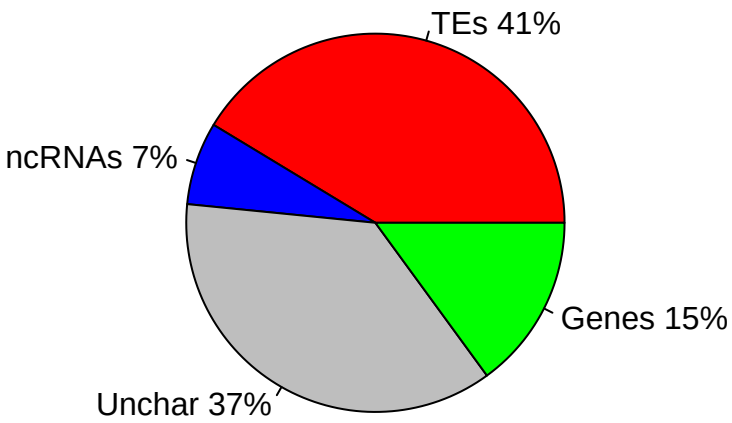
## Visualizing Ping-Pong Hits

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.

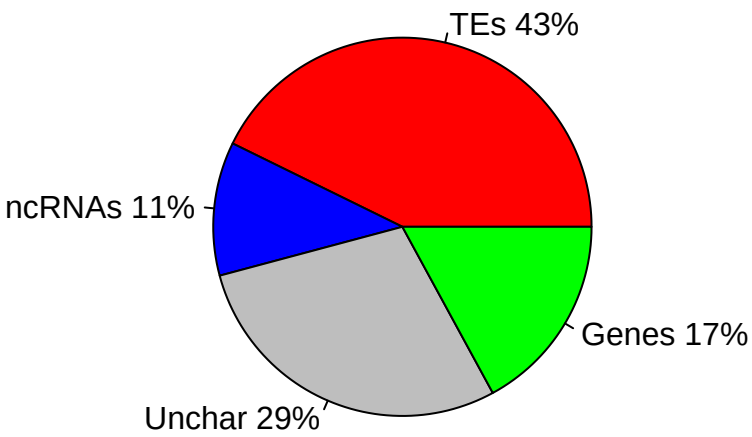
```
Read_Counts_Master_DF <- merge(Read_Counts_Master_DF, Ping_Pong_Matrix, by = "ID", all = T)

#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")
```



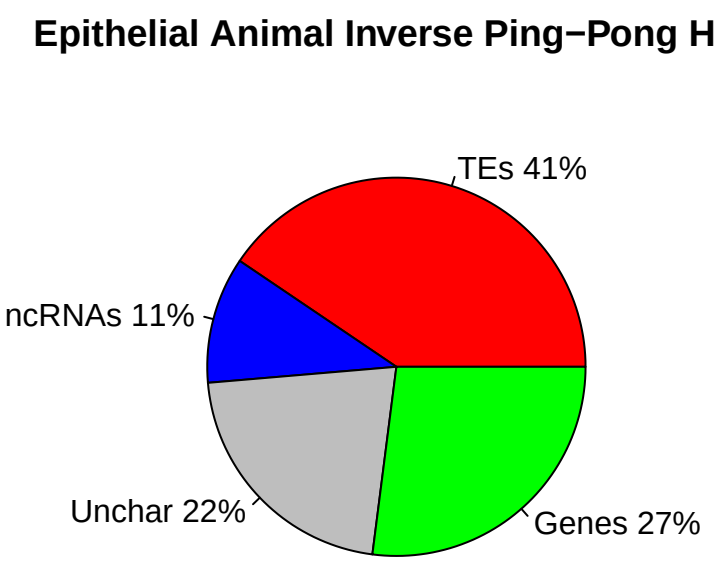

```
#Epithelial Animal Canonical
lbls <- c("TEs", "ncRNAs", "Unchar", "Genes")
colors = c("red", "blue", "gray", "green")
pct <- round(Colch_ping_hits/sum(Colch_ping_hits)*100)
lbls <- paste(lbls, pct) # add percents to labels
lbls <- paste(lbls,"%",sep="") # ad % to labels
pie(Colch_ping_hits,labels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

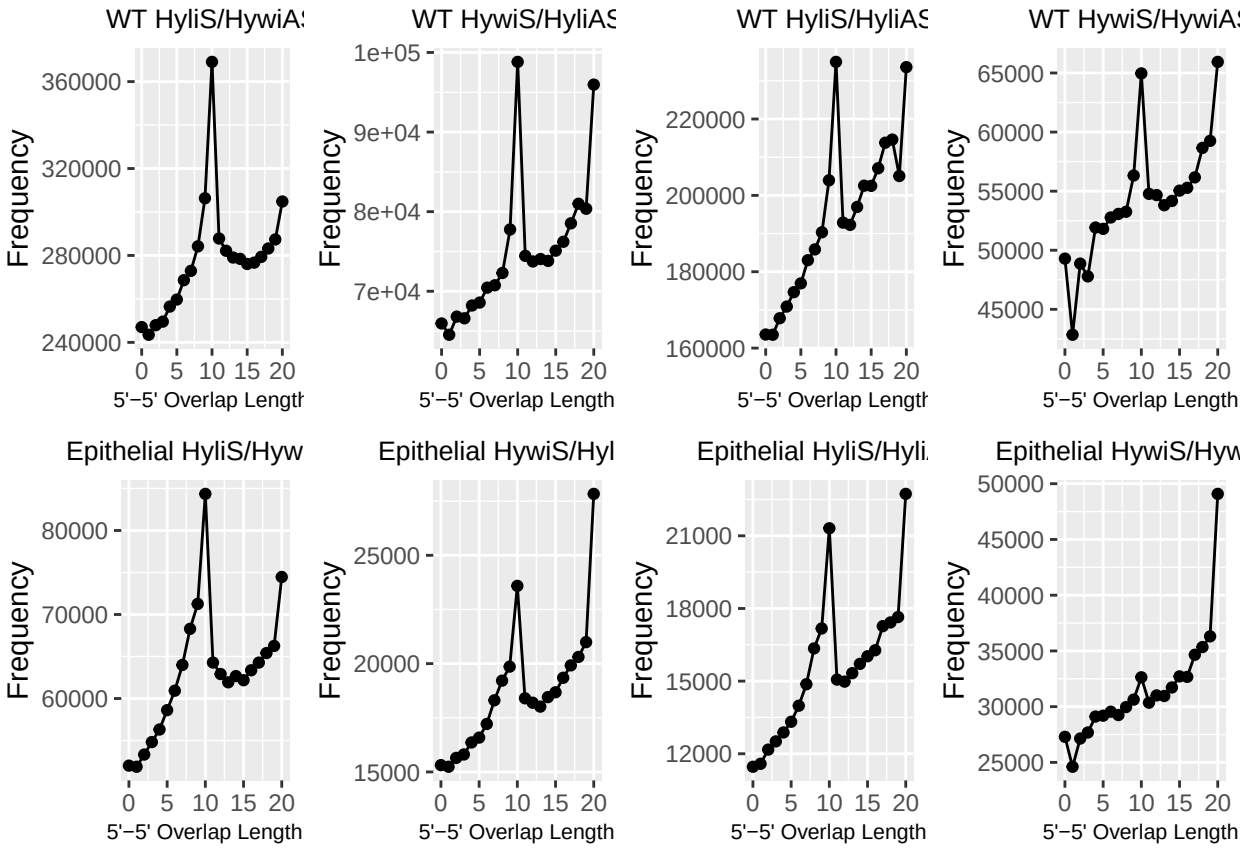
BED files containing piRNA overlap positions and overlap length between pairs of piRNAs (i.e. Hywi antisense/Hyli sense) were generated using the script "windowbed_ping.sh."

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)
```

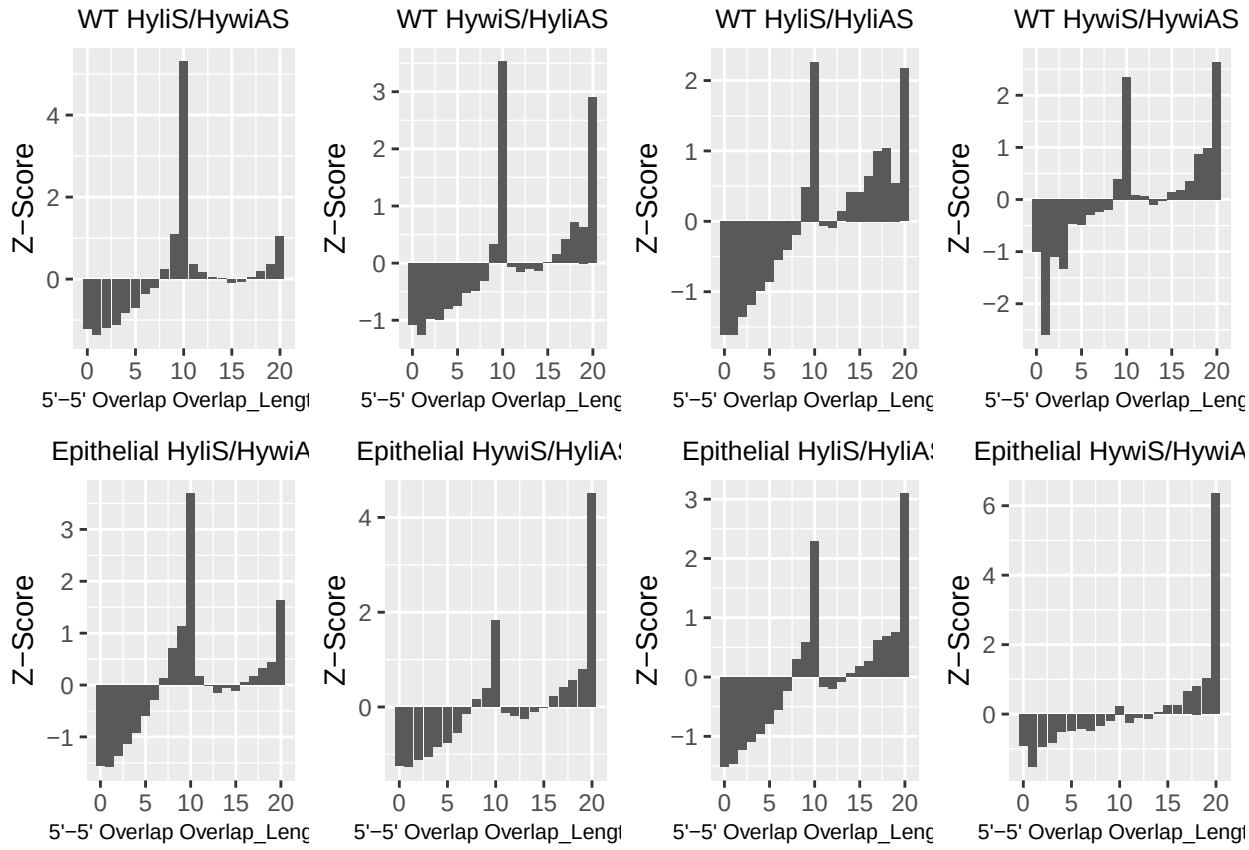



## Z10 scores

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."

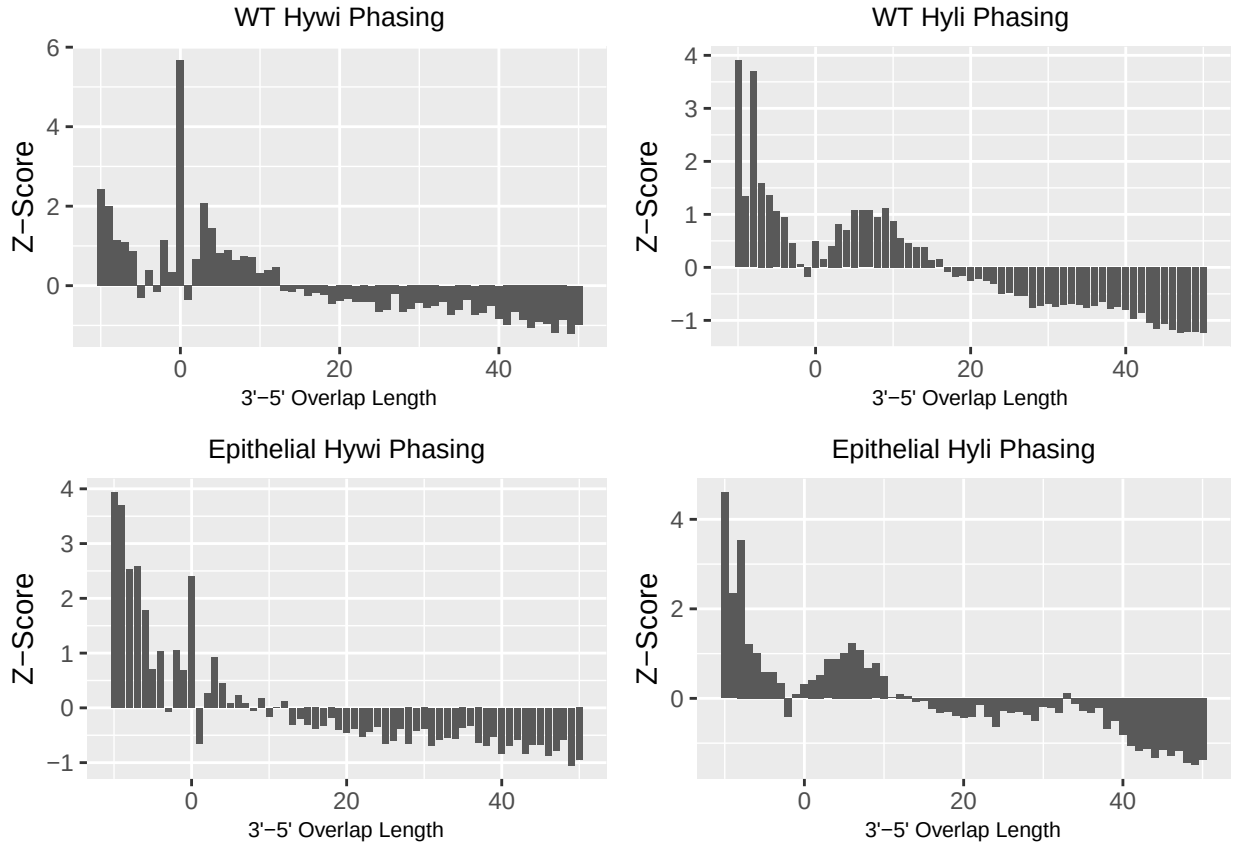
The output of windowBED_phasing.sh are BED files consisting of rows indexing a piRNA overlap event (ex. an overlap event between piRNA_1 and piRNA_2) within a 200 bp window.
The file contains has 11 columns:
1) Transcript on which piRNA_1 is mapped 2) Start position of piRNA_1 3) End position of piRNA_1 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
```