Hierarchical length and sequence preferences establish a single major piRNA 3'-end'

methods by: Daniel Stoyko & Pavol Genzor

12/06/2021

Daniel Stoyko, Pavol Genzor, Astrid D. Haase

This vignette describes the computational materials & methods associated with this manuscript. Please visit **HaaseLab/piRNA__ThreePrimeEnds github repository** to download functions used in the various scripts and analyses. Please refer to the GEO data set **GSE156058** associated with this study for adapter sequences and raw data. The analysis in this vignette was not performed with full data sets, but only subset of the data to demonstrate the materials and methods.

About small RNA Libraries.

Small RNA libraries were prepared and pre-processed as described in Genzor et.al., 2021. Please refer to the corresponding github page for more information.

Pre-requisites & Notes

* Acquire the raw sequencing data from your facility or GEO at NCBI * NOTE: some images may not have rendered perfectly

Vignette Content

Environment setup & test data preparation

Prepare your R working environment by ensuring you have all necessary packages installed and loaded. Download the *.RData* object from *github* to be and load it into R before proceeding.

```r
# load libraries
suppressWarnings(
  suppressPackageStartupMessages({library("data.table"); library("dplyr");
  library("ggplot2"); library("ggpubr"); library("GenomicRanges");
  library("ggseqlogo"); library("BSgenome.Dmelanogaster.UCSC.dm6");
  library("stringr"); library("parallel")}))
```

Load filtered subset of the data

```r
SESSION.DIR="/Users/genzorp/Documents/GITHUB/LIVE/piRNA_ThreePrimeEnds/sample_data/"
load(file = paste0(SESSION.DIR,"single_filtered_brep_1M.RData"))
```

Figure 1: Contribution of piRNA three prime ends

```r
## Data
GR <- original.GR

## Make a table of uniquely mapping piRNAs

uniq.DT <-  as.data.table(GR[mcols(GR)[["NH"]] %in% 1])

## Create a column with 5'-end coordinates
# NOTE: "start" refers to the 5'-end on the "+" strand and
# NOTE: "end" refers to 5`-end on "-" strand

uniq.DT[,FivePrime := paste0(seqnames, "_", ifelse(strand == "-", end, start), "_", strand)]

## Randomly assign each piRNA a value between 1 and N
##    :- N is the number of unique sequences sharing the 5'-end without repeating
##    :- This random value will be used to break ties between equally abundant 3'-ends
##    :- sample() randomly assigns numbers within particular range

uniq.DT[,TieBreaker := base::sample(x = nrow(.SD), size = nrow(.SD), replace = F),
        by = FivePrime]

## Define the Major End as the most abundant sequence per 5'-end
##    :- use TieBreaker value to solve ties in abundance

uniq.DT <- uniq.DT[order(-MULT, TieBreaker)]
uniq.DT[,EndOrder := c(1:nrow(.SD)), by=FivePrime]
uniq.DT[,EndClass := ifelse(EndOrder > 1, "MinorEnds","MajorEnd")]
uniq.DT[["EndClass"]] <- factor(x = uniq.DT[["EndClass"]], levels = c("MinorEnds","MajorEnd"))

## Group 5'-ends into custom bins based on their abundance
##    :- NOTE: the abundance is not normalized here

# summarize abundance
uniq.DT <- setDT(uniq.DT)
uniq.DT[,FivePrimeAbund := sum(MULT), by=FivePrime]

# find bins
aBreaks <- c(1,2,3,4,5,10,50,100,500,1000)
uniq.DT[["aBin"]] <- findInterval(x = uniq.DT[["FivePrimeAbund"]],
                                  vec = aBreaks, left.open = TRUE)

# add and organize bin name
uniq.DT <- setDT(uniq.DT)
uniq.DT[,"binName" := paste0(unique(c(min(FivePrimeAbund),max(FivePrimeAbund))),
                             collapse = "-") , by = aBin]
uniq.DT[FivePrimeAbund > 1000][["binName"]] <- ">1000"
```

```
# order bin names
unique(uniq.DT[["binName"]])
```

```
##  [1] ">1000"   "501-990" "101-500" "51-100"  "11-50"   "6-10"    "5"
##  [8] "4"       "3"       "2"       "1"
```

```
binNameOrder <- c("1","2","3","4","5","6-10","11-50","51-100","101-500","501-999",">1000")
uniq.DT[["aBin"]] <- factor(uniq.DT[["aBin"]])
uniq.DT[["binName"]] <- factor(x = uniq.DT[["binName"]], levels = binNameOrder)

## Calculate contribution of the Major 3'-end to all the ends
uniq.DT <- setDT(uniq.DT)
uniq.DT[,EndContribution := MULT/sum(MULT), by=FivePrime]

## View the generated table
uniq.DT[FivePrime %in% "chr2L_3107310_-"]
```

```
##    seqnames   start     end width strand     N    NH  MULT       FivePrime
##      <fctr>   <int>   <int> <int> <fctr> <int> <int> <int>           <char>
## 1:    chr2L 3107284 3107310    27      -     3     1    27 chr2L_3107310_-
## 2:    chr2L 3107289 3107310    22      -     3     1     3 chr2L_3107310_-
## 3:    chr2L 3107293 3107310    18      -     2     1     2 chr2L_3107310_-
## 4:    chr2L 3107290 3107310    21      -     1     1     1 chr2L_3107310_-
## 5:    chr2L 3107285 3107310    26      -     1     1     1 chr2L_3107310_-
## 6:    chr2L 3107286 3107310    25      -     1     1     1 chr2L_3107310_-
## 7:    chr2L 3107288 3107310    23      -     1     1     1 chr2L_3107310_-
## 8:    chr2L 3107283 3107310    28      -     1     1     1 chr2L_3107310_-
##    TieBreaker EndOrder  EndClass FivePrimeAbund   aBin binName EndContribution
##         <int>    <int>    <fctr>          <int> <fctr>  <fctr>           <num>
## 1:          5        1  MajorEnd             11      6   11-50      0.72972973
## 2:          1        2 MinorEnds             11      6   11-50      0.08108108
## 3:          7        3 MinorEnds             18      6   11-50      0.05405405
## 4:          2        4 MinorEnds             18      6   11-50      0.02702703
## 5:          3        5 MinorEnds             18      6   11-50      0.02702703
## 6:          4        6 MinorEnds             18      6   11-50      0.02702703
## 7:          6        7 MinorEnds             15      6   11-50      0.02702703
## 8:          8        8 MinorEnds             15      6   11-50      0.02702703
```

```
## Prepare data for visualization
Barplot.DT <- uniq.DT[, .(Sum_Abundance = sum(MULT)), by=c("EndClass","aBin","binName")]
Barplot.DT
```

```
##      EndClass   aBin binName Sum_Abundance
##        <fctr> <fctr>  <fctr>         <int>
##   1:  MajorEnd     10   >1000        310025
##   2: MinorEnds     10   >1000        188991
##   3:  MajorEnd      9    <NA>         76192
##   4: MinorEnds      9    <NA>         38932
##   5:  MajorEnd      8 101-500        171522
##   6: MinorEnds      8 101-500        104056
##   7:  MajorEnd      7  51-100         97807
##  8: MinorEnds      7  51-100         49779
##  9:  MajorEnd      6   11-50        308227
## 10: MinorEnds      6   11-50        134920
## 11:  MajorEnd      5    6-10        203979
## 12: MinorEnds      5    6-10         58833
## 13:  MajorEnd      4       5         77908
## 14: MinorEnds      4       5         18856
## 15:  MajorEnd      3       4        112129
## 16: MinorEnds      3       4         24238
## 17:  MajorEnd      2       3        234059
## 18: MinorEnds      2       3         52074
## 19:  MajorEnd      1       2        374588
## 20: MinorEnds      1       2         40532
## 21:  MajorEnd      0       1        312904
## 22: MinorEnds      0       1         13089
##      EndClass   aBin binName Sum_Abundance
```

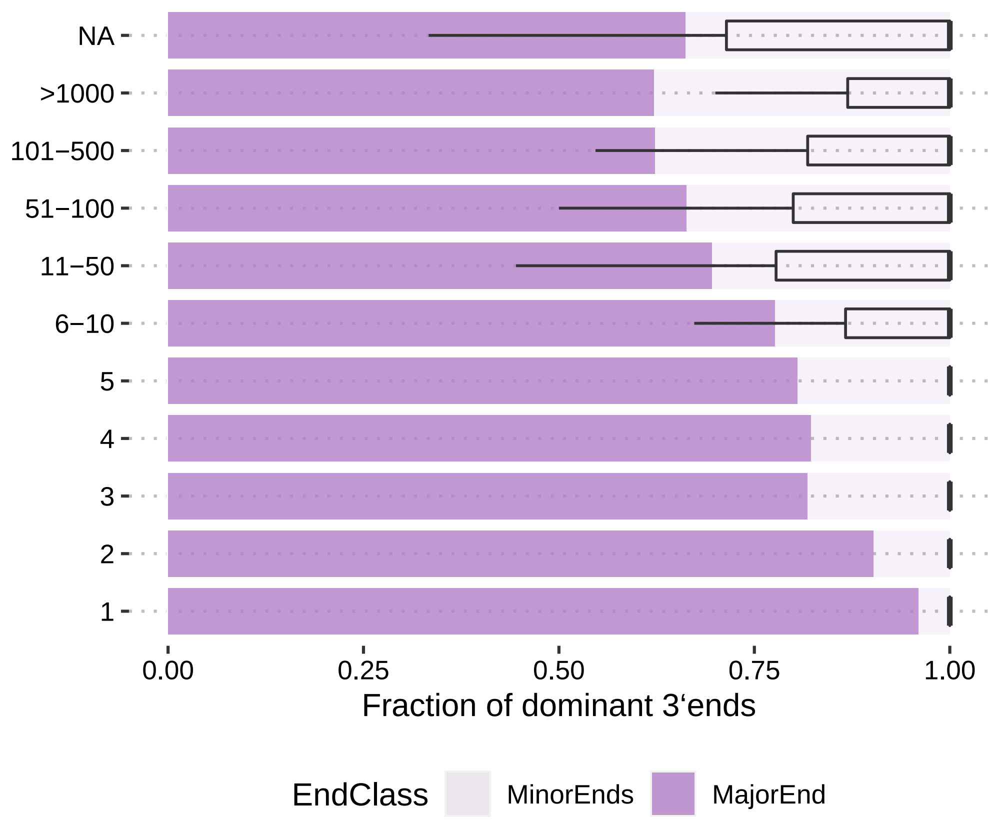
```
NumberOfEnds.DT <- uniq.DT[, .N, by=c("FivePrime", "aBin","binName")]
NumberOfEnds.DT
```

```
##                FivePrime   aBin binName     N
##                   <char> <fctr>  <fctr> <int>
##      1: chr2L_16698758_+     10   >1000    10
##      2: chr2L_16697936_+     10   >1000     9
##      3:  chr2L_7425853_+     10   >1000     7
##      4:  chr2L_7425979_+     10   >1000     9
##      5: chr2L_16698418_+     10   >1000     7
##     ---
## 560082:  chr2R_5199715_-      5    6-10     1
## 560083: chr2L_19466782_+      5    6-10     1
## 560084:  chr2R_5178183_-      6   11-50     1
## 560085: chr2L_10214356_-      7  51-100     1
## 560086: chr2L_19466575_+      9    <NA>     1
```

```
MeanNumberOfEnds.DT <- NumberOfEnds.DT[, lapply(.SD,mean), by=c("aBin","binName"), .SDcols="N"]
MeanNumberOfEnds.DT
```

```
##       aBin binName        N
##     <fctr>  <fctr>    <num>
##  1:     10   >1000 2.084677
##  2:      9    <NA> 2.140146
##  3:      8 101-500 1.926704
##  4:      7  51-100 1.706522
##  5:      6   11-50 1.576287
##  6:      5    6-10 1.301198
##  7:      4       5 1.199211
##  8:      3       4 1.189426
##  9:      2       3 1.256939
## 10:      1       2 1.114937
## 11:      0       1 1.052958
```

```r
##  NOTE: Using the three tables generated in previous chunk
#       Barplot.DT
#       NumberOfEnds.DT
#       MeanNumberOfEnds.DT

##
EndContributionPlot <- ggplot() + theme_pubclean() +
  geom_bar(data = Barplot.DT,
           aes(x = binName, y = Sum_Abundance, alpha = EndClass, group = EndClass),
           stat = "identity", position = "fill", fill="#B37FC7",
           width = 0.8) +
  geom_boxplot(data = uniq.DT[EndClass %in% "MajorEnd"],
               aes(x = binName, y= EndContribution),
               outlier.shape = NA, width = 0.5, fill = NA) +
  ylab("Fraction of dominant 3`ends") + xlab("") + coord_flip() +
  scale_alpha_manual(values=c(0.1,0.8)) +
  theme(legend.position="bottom",
        axis.text = element_text(family = "Helvetica",colour = "black", size = 10),
        aspect.ratio=0.75); EndContributionPlot
```



```r
##
NumberOfEndsPlot <- ggplot() + theme_pubclean() +
  geom_boxplot(data = NumberOfEnds.DT,
               aes(x = binName, y = N),
               color="black", fill="#C49CD3",
               outlier.shape = NA) +
  geom_point(data = MeanNumberOfEnds.DT,
             aes(x= binName, y = N),
             shape = 16, color = "firebrick3", size = 6) +
  ylab("number of unique 3`ends") + xlab("") +
  ggtitle(paste0("red = mean")) + coord_flip() +
  theme(aspect.ratio = 0.75,
        axis.text = element_text(family = "Helvetica",colour = "black", size = 10),
        legend.position = "none"); NumberOfEndsPlot
```

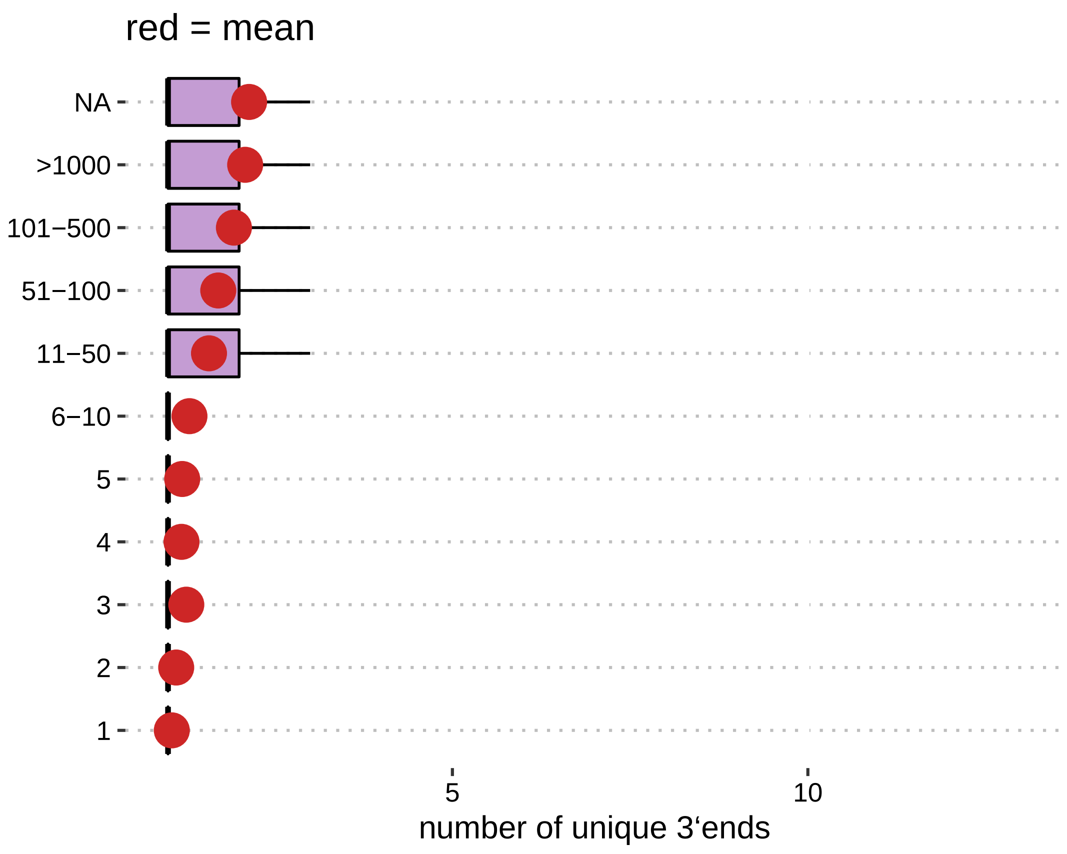


Figure 2: Size distribution of major and minor ends of piRNAs

```
## NOTE: Use that table generated in the previous chunk
##    uniq.DT

## Table by end type
MajorMinorLengths.DT <- uniq.DT[,lapply(.SD,sum), by=c("EndClass","width"), .SDcols="MULT"]
MajorMinorLengths.DT <- MajorMinorLengths.DT[order(EndClass,width)]
MajorMinorLengths.DT[,"groupReadSum" := sum(.SD),by="EndClass", .SDcols="MULT"]
MajorMinorLengths.DT[,"totalReads" := sum(.SD), .SDcols="MULT"]
MajorMinorLengths.DT[,"groupPercent" := (MULT/groupReadSum)*100]
MajorMinorLengths.DT[,"totalPercent" := (MULT/totalReads)*100]
MajorMinorLengths.DT
```

```
##      EndClass width   MULT groupReadSum totalReads groupPercent totalPercent
##        <fctr> <int>  <int>        <int>      <int>        <num>        <num>
##   1: MinorEnds    18  14295       724300    3003640 1.973630e+00 4.759225e-01
##  2: MinorEnds    19  17585       724300    3003640 2.427861e+00 5.854563e-01
##  3: MinorEnds    20  29861       724300    3003640 4.122739e+00 9.941604e-01
##  4: MinorEnds    21  58641       724300    3003640 8.096231e+00 1.952331e+00
##  5: MinorEnds    22  83027       724300    3003640 1.146307e+01 2.764213e+00
##  6: MinorEnds    23  48168       724300    3003640 6.650283e+00 1.603654e+00
##  7: MinorEnds    24 116744       724300    3003640 1.611818e+01 3.886751e+00
##  8: MinorEnds    25 123195       724300    3003640 1.700884e+01 4.101523e+00
##  9: MinorEnds    26 116621       724300    3003640 1.610120e+01 3.882656e+00
## 10: MinorEnds    27  99043       724300    3003640 1.367431e+01 3.297432e+00
## 11: MinorEnds    28  16153       724300    3003640 2.230153e+00 5.377808e-01
## 12: MinorEnds    29    857       724300    3003640 1.183211e-01 2.853205e-02
## 13: MinorEnds    30     90       724300    3003640 1.242579e-02 2.996364e-03
## 14: MinorEnds    31     17       724300    3003640 2.347094e-03 5.659799e-04
## 15: MinorEnds    32      3       724300    3003640 4.141930e-04 9.987881e-05
## 16:  MajorEnd    18  40386      2279340    3003640 1.771829e+00 1.344569e+00
## 17:  MajorEnd    19  32049      2279340    3003640 1.406065e+00 1.067005e+00
## 18:  MajorEnd    20  28716      2279340    3003640 1.259838e+00 9.560400e-01
## 19:  MajorEnd    21  67803      2279340    3003640 2.974677e+00 2.257361e+00
## 20:  MajorEnd    22  77983      2279340    3003640 3.421297e+00 2.596283e+00
## 21:  MajorEnd    23 118871      2279340    3003640 5.215150e+00 3.957565e+00
## 22:  MajorEnd    24 305666      2279340    3003640 1.341029e+01 1.017652e+01
## 23:  MajorEnd    25 720392      2279340    3003640 3.160529e+01 2.398397e+01
## 24:  MajorEnd    26 674169      2279340    3003640 2.957738e+01 2.244507e+01
## 25:  MajorEnd    27 196817      2279340    3003640 8.634824e+00 6.552616e+00
## 26:  MajorEnd    28  15607      2279340    3003640 6.847158e-01 5.196029e-01
## 27:  MajorEnd    29    738      2279340    3003640 3.237779e-02 2.457019e-02
## 28:  MajorEnd    30     88      2279340    3003640 3.860767e-03 2.929779e-03
## 29:  MajorEnd    31     47      2279340    3003640 2.062000e-03 1.564768e-03
## 30:  MajorEnd    32      8      2279340    3003640 3.509788e-04 2.663435e-04
##      EndClass width   MULT groupReadSum totalReads groupPercent totalPercent
```

```r
## All reads
AllLengths.DT <- uniq.DT[,.(MULT = sum(.SD)), by=c("width"), .SDcols="MULT"]
AllLengths.DT[,"totalReads" := sum(.SD), .SDcols = "MULT"]
AllLengths.DT[,"totalPercent" := (MULT/totalReads)*100]
AllLengths.DT
```

```
##     width   MULT totalReads totalPercent
##     <int>  <int>      <int>        <num>
##  1:    23 167039    3003640 5.561219e+00
##  2:    22 161010    3003640 5.360496e+00
##  3:    24 422410    3003640 1.406327e+01
##  4:    21 126444    3003640 4.209692e+00
##  5:    25 843587    3003640 2.808549e+01
##  6:    26 790790    3003640 2.632772e+01
##  7:    20  58577    3003640 1.950200e+00
##  8:    27 295860    3003640 9.850049e+00
##  9:    19  49634    3003640 1.652462e+00
## 10:    18  54681    3003640 1.820491e+00
## 11:    28  31760    3003640 1.057384e+00
## 12:    29   1595    3003640 5.310224e-02
## 13:    30    178    3003640 5.926143e-03
## 14:    31     64    3003640 2.130748e-03
## 15:    32     11    3003640 3.662223e-04
## Plot
LengthDistributionPlot <- ggplot() + theme_pubclean() +
  geom_bar(data = AllLengths.DT,
           aes(x=width, y=totalPercent),
           stat="identity", fill="gray80", width = 0.8) +
  geom_line(data = MajorMinorLengths.DT,
          aes(x = width, y = groupPercent, colour = EndClass),
          size = 1) +
  geom_point(data = MajorMinorLengths.DT,
           aes(x = width, y = groupPercent, shape = EndClass, colour = EndClass),
           size = 4) +
  xlab("size (nt)") + ylab("percent of piRNAs") +
  scale_x_continuous(breaks = seq(0,40,2)) +
  scale_y_continuous(breaks = seq(0,40,5)) +
  scale_colour_manual(values = c("#AB72C0","#FF6D33")) +
  theme(aspect.ratio = 1, legend.position = "top",
      axis.text = element_text(family = "Helvetica", colour = "black", size = 10))

LengthDistributionPlot
```

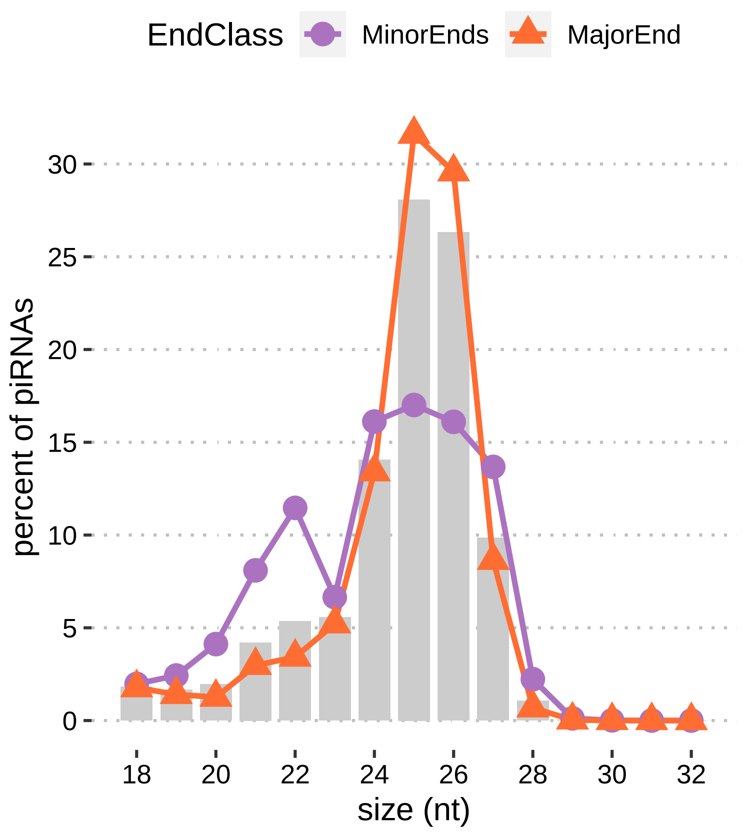


Figure S2: Distance between major ends in WT and PNLDC KO

```
## NOTE: Scripts used for comparing mouse WT to PNLDC KO piRNA major ends
## NOTE: Sample data is not provided. Data can be obtained from PRJNA421205
## See methods for details
## Set GR1 as the WT library
## Set GR2 as the PNLDC1 KO library

GR1 <- MOUSE.BAM.GR.WT
GR2 <- MOUSE.BAM.GR.KO

## For GR1, determine the major 3'-end in same manner as described previously
uniq.DT1 <-  as.data.table(GR1[mcols(GR1)[["NH"]] %in% 1])
uniq.DT1[,FivePrime1 := paste0(seqnames, "_", ifelse(strand == "-", end, start), "_", strand)]
uniq.DT1[,TieBreaker := base::sample(x = nrow(.SD), size = nrow(.SD), replace = F),
        by = FivePrime1]
uniq.DT1 <- uniq.DT1[order(-MULT, TieBreaker)]
uniq.DT1[,EndOrder := c(1:nrow(.SD)), by=FivePrime1]
uniq.DT1[,EndClass := ifelse(EndOrder > 1, "MinorEnds","MajorEnd")]

## keep only major 3'-ends
uniq.DT1.maj <- uniq.DT1[EndClass %in% c("MajorEnd")]

## Determine 3'-end coordinate and discard unnecessary columns
uniq.DT1.maj[,ThreePrime1 := ifelse(strand == "-", start, end)]
uniq.DT1.maj <- uniq.DT1.maj[,c(5, 7, 8, 12)]
colnames(uniq.DT1.maj) <- c( "strand1", "MULT1", "FivePrime1", "ThreePrime1")

uniq.DT1.maj

## Repeat for GR2
uniq.DT2 <-  as.data.table(GR2[mcols(GR2)[["NH"]] %in% 1])
uniq.DT2[,FivePrime2 := paste0(seqnames, "_", ifelse(strand == "-", end, start), "_", strand)]
uniq.DT2[,TieBreaker := base::sample(x = nrow(.SD), size = nrow(.SD), replace = F),
        by = FivePrime2]
uniq.DT2 <- uniq.DT2[order(-MULT, TieBreaker)]
uniq.DT2[,EndOrder := c(1:nrow(.SD)), by=FivePrime2]
uniq.DT2[,EndClass := ifelse(EndOrder > 1, "MinorEnds","MajorEnd")]
uniq.DT2.maj <- uniq.DT2[EndClass %in% c("MajorEnd")]
uniq.DT2.maj[,ThreePrime2 := ifelse(strand == "-", start, end)]
uniq.DT2.maj <- uniq.DT2.maj[,c(5, 7, 8, 12)]
colnames(uniq.DT2.maj) <- c("strand2", "MULT2", "FivePrime2", "ThreePrime2")

## Merge the two files by 5'-end coordinate
## Discard sequences with 5'-ends not present in both datasets
DT.combined <- merge(uniq.DT1.maj, uniq.DT2.maj, by.x="FivePrime1", by.y="FivePrime2", all.x=F, all.y =

## Calculate total abundance of piRNAs in a pair
DT.combined[, MULTtotal := MULT1 + MULT2]

## Determine distance between the two major 3'-ends
DT.combined[, Distance := ifelse(strand1 == "+", ThreePrime2-ThreePrime1, ThreePrime1-ThreePrime2)]

## Keep sequences where PNLDC KO piRNA is longer than the WT piRNA
DT.combined <- DT.combined[Distance > 0]

## Count number of reads for each distance
DT.combined.sum <- DT.combined[, lapply(.SD, sum), .SDcols="MULTtotal", by="Distance"]
```

```r
## Convert to % of total
DT.combined.sum$MULTtotal <- (DT.combined.sum$MULTtotal/ sum(DT.combined.sum$MULTtotal))*100

## Plot
DistancePlot <- ggplot() + theme_bw() +
  geom_bar(data=DT.combined.sum,
           aes(x=Distance, y = MULTtotal),
           color= "black",
           stat="identity",
           width= 0.85) +
  labs(title = "SampleData",
       x= "Distance between the major 3'ends in WT and PNLDC KO",
       y= "% of total") +
  theme(panel.grid = element_blank(),
        aspect.ratio = 1)
```

Figure 3: Major and minor -dependent nucleotide frequencies of piRNAs

```r
## Data
GR <- original.GR

## Use only uniquely mapping piRNAs
uniq.GR <- GR[mcols(GR)[["NH"]] %in% "1"]
uniq.DT <- as.data.table(uniq.GR)

## Expand the GR to include surrounding nucleotides
uniq.GRE <- uniq.GR
end(uniq.GRE) <- end(uniq.GRE) + 4
start(uniq.GRE) <- start(uniq.GRE) - 4

## Add sequence to the table
uniq.DT[["seq"]] <- as.vector(getSeq(BSgenome.Dmelanogaster.UCSC.dm6, uniq.GRE))

## Create a column with 5'-end coordinates
uniq.DT <- setDT(uniq.DT)
uniq.DT[,FivePrime := paste0(seqnames, "_", ifelse(strand == "-", end, start), "_", strand)]

## Randomly assign each piRNA a value between 1 and N
uniq.DT[,TieBreaker := sample(nrow(.SD), size = nrow(.SD), replace = F), by = FivePrime]

## Define the Major End as the most abundant sequence per 5'-end
uniq.DT <- uniq.DT[order(-MULT, TieBreaker)]
uniq.DT[,EndOrder := c(1:nrow(.SD)), by=FivePrime]
uniq.DT[,EndClass := ifelse(EndOrder > 1, "MinorEnds","MajorEnd")]
uniq.DT[["EndClass"]] <- factor(x = uniq.DT[["EndClass"]], levels = c("MinorEnds","MajorEnd"))

## Group 5'-ends into custom bins based on their abundance
uniq.DT <- setDT(uniq.DT)
uniq.DT[,FivePrimeAbund := sum(MULT), by=FivePrime]
uniq.DT
```

```
##          seqnames    start      end width strand     N    NH   MULT
##           <fctr>    <int>    <int> <int> <fctr> <int> <int> <int>
##      1:    chr2L 16698758 16698780    23      +     3     1 28686
##      2:    chr2L 16697936 16697957    22      +     3     1 27064
##      3:    chr2L 16698758 16698779    22      +     3     1 26662
##      4:    chr2L  7425853  7425875    23      +     3     1 24812
##      5:    chr2L  7425979  7426002    24      +     3     1 23638
##     ---
## 666754:    chr2L 19467210 19467238    29      +     1     1     1
## 666755:    chr2L 19467405 19467432    28      +     1     1     1
## 666756:    chr2R  5050923  5050953    31      +     1     1     1
## 666757:    chr2R  5055725  5055744    20      +     1     1     1
## 666758:    chr2R  5055342  5055370    29      +     1     1     1
##                                          seq         FivePrime TieBreaker
##                                       <char>            <char>      <int>
##      1:         TTGCTCTTTGGTGATTTTAGCTGTATGGTGT chr2L_16698758_+          7
##      2:          TGTTTCTTTGGTATTCTAGCTGTAGATTGT chr2L_16697936_+          9
##      3:          TTGCTCTTTGGTGATTTTAGCTGTATGGTG chr2L_16698758_+          6
##      4:         ACAGTCAGGTACCTGAAGTAGCGCGCGTGGT  chr2L_7425853_+          2
##      5:        GTCTATTGTACTTCATCAGGTGCTCTGGTGTG  chr2L_7425979_+          7
##     ---
## 666754:   ACGCTTATTTGTTGATTAGTTCTAGCCTTAGTTTCCC chr2L_19467210_+         10
## 666755:    AAGTTAAAACACCGCAAGCTGGAAGAACCGATGTAT chr2L_19467405_+         10
## 666756: CAACTTACGCATATGTGAGTGGGGAAAGGACTCGGACAG  chr2R_5050923_+         10
## 666757:             AATTTGTTTCGTCAACGTATGCAATATT  chr2R_5055725_+         10
## 666758:   TAAATCTTGATTTGCGGTGCTTCCACCTGCAAACTCT  chr2R_5055342_+         11
##         EndOrder  EndClass FivePrimeAbund
##            <int>    <fctr>          <int>
##      1:        1  MajorEnd          67950
##      2:        1  MajorEnd          41154
##      3:        2 MinorEnds          67950
##      4:        1  MajorEnd          46882
##      5:        1  MajorEnd          26062
##     ---
## 666754:       11 MinorEnds            277
## 666755:       10 MinorEnds            161
## 666756:       11 MinorEnds            150
## 666757:       10 MinorEnds            108
## 666758:       12 MinorEnds            270
```

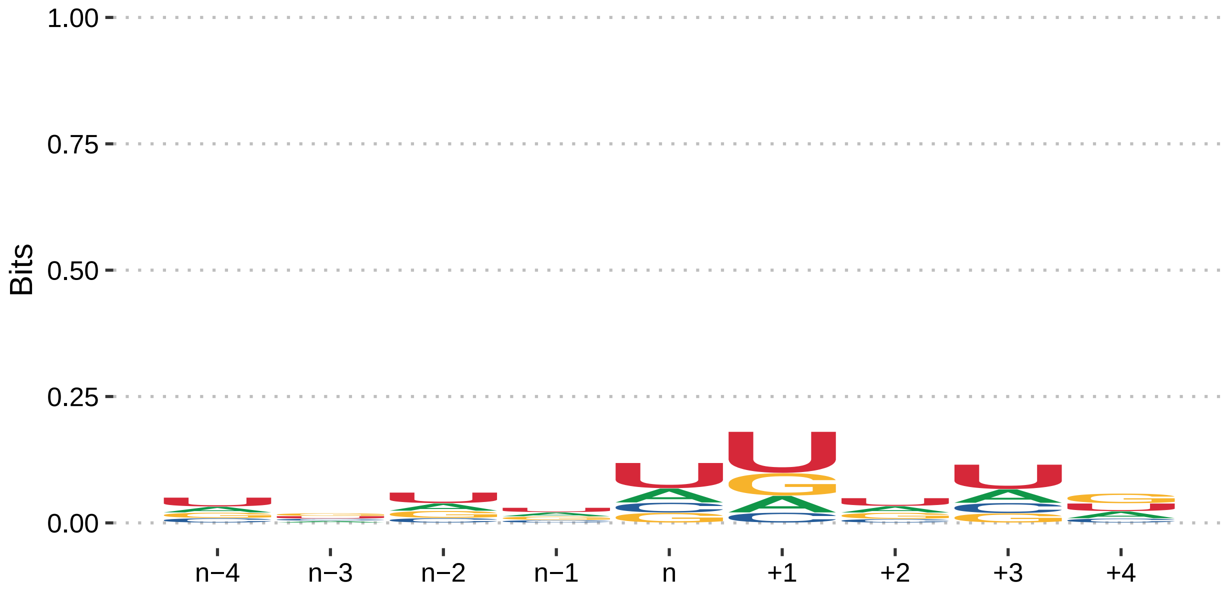
```
## View the generated table
uniq.DT[FivePrime %in% "chr2L_3107310_-"]
```

```
##    seqnames   start     end width strand     N    NH  MULT
##      <fctr>   <int>   <int> <int> <fctr> <int> <int> <int>
## 1:    chr2L 3107284 3107310    27      -     3     1    27
## 2:    chr2L 3107289 3107310    22      -     3     1     3
## 3:    chr2L 3107293 3107310    18      -     2     1     2
## 4:    chr2L 3107290 3107310    21      -     1     1     1
## 5:    chr2L 3107285 3107310    26      -     1     1     1
## 6:    chr2L 3107283 3107310    28      -     1     1     1
## 7:    chr2L 3107288 3107310    23      -     1     1     1
## 8:    chr2L 3107286 3107310    25      -     1     1     1
##                                  seq       FivePrime TieBreaker EndOrder
##                               <char>          <char>      <int>    <int>
## 1:  CAACTCTCTCTGCGTCTCTCTATAGACCCGATTAC chr2L_3107310_-          7        1
## 2:       CAACTCTCTCTGCGTCTCTCTATAGACCCG chr2L_3107310_-          5        2
## 3:           CAACTCTCTCTGCGTCTCTCTATAGA chr2L_3107310_-          8        3
## 4:        CAACTCTCTCTGCGTCTCTCTATAGACCC chr2L_3107310_-          1        4
## 5:   CAACTCTCTCTGCGTCTCTCTATAGACCCGATTA chr2L_3107310_-          2        5
## 6: CAACTCTCTCTGCGTCTCTCTATAGACCCGATTACC chr2L_3107310_-          3        6
## 7:      CAACTCTCTCTGCGTCTCTCTATAGACCCGA chr2L_3107310_-          4        7
## 8:    CAACTCTCTCTGCGTCTCTCTATAGACCCGATT chr2L_3107310_-          6        8
##     EndClass FivePrimeAbund
##       <fctr>          <int>
## 1:  MajorEnd             16
## 2: MinorEnds             16
## 3: MinorEnds             10
## 4: MinorEnds             10
## 5: MinorEnds             10
## 6: MinorEnds             10
## 7: MinorEnds             11
## 8: MinorEnds             11
```

```
## Plotting LOGO
##  1. select type of end and its sequences
##  2. extract sub-string
##  3. expand to reads
##  4. replace nucleotides

ForLogo.second.DT <- uniq.DT[EndOrder %in% 2][,c("MULT", "seq")]
ForLogo.second.DT[,Seq_Range := substring(seq, nchar(seq)-8 ,nchar(seq))]
ForLogo.second.DT <- ForLogo.second.DT[ rep( c(1:nrow(ForLogo.second.DT)),
                                             times = MULT)]
ForLogo.second.DT[["Seq_Range"]] <- gsub("T","U",ForLogo.second.DT[["Seq_Range"]])

##
## Plot LOGO
##

LogoPlot <- ggplot() + theme_pubclean() +
  geom_logo(data=ForLogo.second.DT[["Seq_Range"]],
            method="bits", seq_type="rna") +
  scale_y_continuous(limits = c(0,1)) +
  scale_x_continuous(
    breaks = 1:9,
    labels = paste(c("n-4","n-3","n-2","n-1","n","+1","+2","+3","+4")) ) +
  theme(panel.grid = element_blank(),
        axis.text = element_text(family = "Helvetica", colour = "black", size = 10),
        aspect.ratio = 0.5); LogoPlot
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```

```
## Scale for 'x' is already present. Adding another scale for 'x', which will
## replace the existing scale.
```



```
##
## Plot +1 nucleotide frequency
##


## For barplots with +1 nucleotide composition:
## isolate the +1 nucleotide

uniq.DT[, plusOne := substring(seq, nchar(seq)-3, nchar(seq)-3)]
PlusOneComposition.DT <- uniq.DT[, .(MULT = sum(MULT)), by=c("EndClass","plusOne")]
PlusOneComposition.DT[["plusOne"]] <- factor(PlusOneComposition.DT[["plusOne"]], c("A","G","C","T"))

## set colors
nuc_colors <- c("#00AF54", "#FFD639", "#447EC5", "#DF2935")

## plot
PlusOneCompositionPlot <- ggplot() + theme_pubclean()+
  geom_bar(data = PlusOneComposition.DT,
           aes(x = EndClass, y = MULT, fill = plusOne),
           stat="identity", position="fill", width = 0.8) +
  scale_fill_manual(values=nuc_colors) +
  ylab("fraction nucleotide") +
  theme(aspect.ratio = 2,
        legend.position = "right",
        axis.text = element_text(family = "Helvetica",colour = "black", size = 10),
        panel.grid = element_blank()); PlusOneCompositionPlot
```

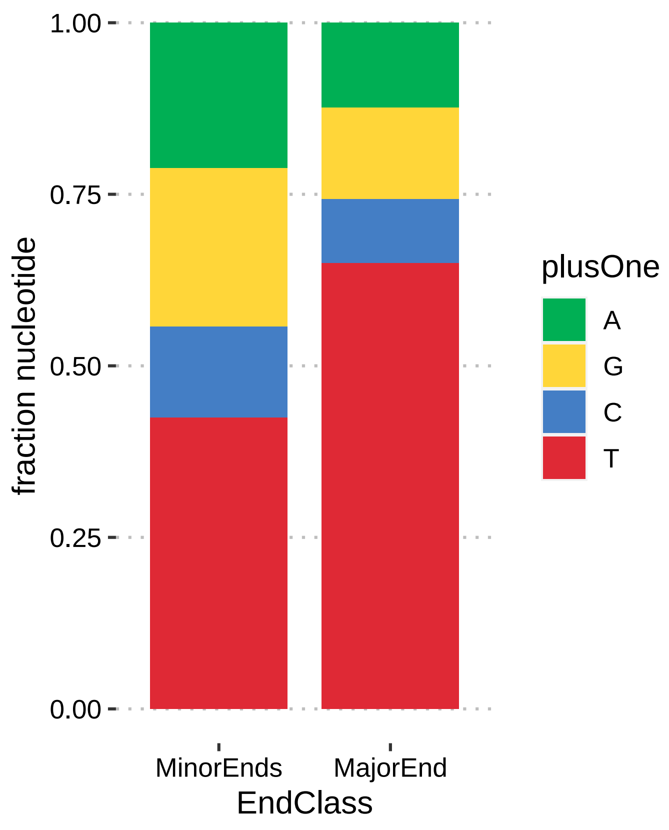


Figure 4 Heatmaps: Distribution of 3'-ends relative to 5'-ends

```
## Crate main objects

GR <- original.GR
uniq.GR <- GR[mcols(GR)[["NH"]] %in% 1]
uniq.DT <- as.data.table(uniq.GR)

## Extend the 3`-end of piRNAs so all piRNAs are 50-nt long & add sequence to the list

uniq.GRE <- uniq.GR
end(uniq.GRE[strand(uniq.GRE)=="+"]) <- start(uniq.GRE[strand(uniq.GRE)=="+"])+50
start(uniq.GRE[strand(uniq.GRE)=="-"]) <- end(uniq.GRE[strand(uniq.GRE)=="-"])-50
uniq.DT[["seq"]] <- as.vector(getSeq(BSgenome.Dmelanogaster.UCSC.dm6, uniq.GRE))

## Set analysis range, trim reads to this range & replace T's for U's
aRange <- c(18:32)
uniq.DT[,seq := substring(seq, min(aRange), max(aRange))]
uniq.DT[["seq"]] <- gsub("T", "U", uniq.DT[["seq"]])

## Create a vector of sequence contexts you wish to analyze. Must be same length as range above.
## Add any custom contexts you would like to use
##    - shorthand symbols:
##      N: U, C, G, or A
##      V: C, G, or A (no U)
##      i.e.: "NUNNNNNNNNNNNNN"
##      This sequence will be used to select have a U 19-nt away from the 5'-end
```

```r
aContexts <- unlist(lapply(c(0: (length(aRange)-1)), function(i){
   paste(c(rep("N", i), "U", rep("N",(length(aRange)-1)-i)), collapse="")}))
customContexts <- c("NNNNNNNNNNNNNNN","NNNNNNVVVVVVNNN", "NNNNNNVVVVVVUNN", "NNNNNUVVVVVVNNN")
aContexts <- c(aContexts, customContexts)


## Calculate size distribution for all piRNAs
## Make sure all lengths in the window of analysis are represented in the Distribution
## Normalize by converting to fraction / percentage of total

DistributionOfAll <- uniq.DT[, .(MULT = sum(MULT)), by="width"][order(width)]
if(any(!aRange %in% DistributionOfAll[["width"]])){
    DistributionOfAll <- bind_rows(DistributionOfAll,
                                   data.table(width = setdiff(aRange, DistributionOfAll[["width"]]),
                                              MULT = 0))}
DistributionOfAll[,"Percent" := (MULT/sum(MULT))*100]


## Determine the piRNA 3'-end distribution for each context

a_context <- aContexts[1]
Regex <- gsub("N", "[UCGA]", a_context)

## Loop through contexts
EndDistribution <- rbindlist(lapply(seq_along(aContexts), function(a){
  # Start and report progress
  message(paste0("a context: ",a))
  a_context <- aContexts[a]

  # Convert context to regex format and simplify
  a_regex_format <- gsub("N", "[UCGA]", a_context)
  a_regex_format <- gsub("V", "[CGA]", a_regex_format)

  # Find sequences which match the context and determine size distribution
  subset.DT <- uniq.DT[str_detect(seq, a_regex_format)][, .(MULT = sum(MULT)), by="width"]

  # Make sure all lengths in the total distribution are represented in the subset
  if(any(!DistributionOfAll[["width"]] %in% subset.DT[["width"]])){
    subset.DT <- bind_rows(subset.DT,
                           data.table(width = setdiff(DistributionOfAll[["width"]],
                                                      subset.DT[["width"]]), MULT = 0))}

  # Make sure all lengths in the window of analysis are represented in the subset
  if(any(!aRange %in% subset.DT[["width"]])){
    subset.DT <- bind_rows(subset.DT,
                           data.table(width = setdiff(aRange,
                                                      subset.DT[["width"]]), MULT = 0))}

  # order and normalize, and find deviation
  subset.DT <- subset.DT[order(width)]
  subset.DT[,"Percent" := (MULT/sum(MULT))*100]
  subset.DT[,"DeviationFromNormalDistribution" := Percent - DistributionOfAll[["Percent"]]]
  # Add data identifiers
  subset.DT[["ContextNumber"]] <- a
  subset.DT[["Seq"]] <- strsplit(a_context, "")[[1]]
  return(subset.DT) }))
```

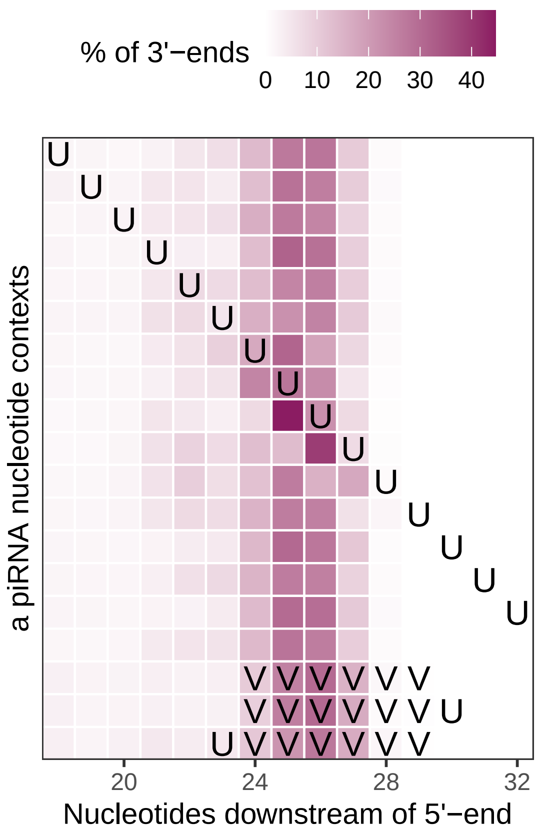
```
## a context: 1

## a context: 2

## a context: 3

## a context: 4

## a context: 5

## a context: 6

## a context: 7

## a context: 8

## a context: 9

## a context: 10

## a context: 11

## a context: 12

## a context: 13

## a context: 14

## a context: 15

## a context: 16

## a context: 17

## a context: 18

## a context: 19
```

```
EndDistribution
```

```
##      width  MULT     Percent DeviationFromNormalDistribution ContextNumber
##      <int> <num>       <num>                           <num>         <int>
##   1:    18 18915 2.059815787                     0.239324650             1
##   2:    19 18430 2.006999998                     0.354538318             1
##   3:    20 13728 1.494959087                    -0.455241337             1
##   4:    21 24285 2.644600920                    -1.565091321             1
##   5:    22 45587 4.964357510                    -0.396138422             1
##  ---
## 281:    28  1112 1.932434311                     0.875050603            19
## 282:    29    89 0.154664257                     0.101562021            19
## 283:    30    10 0.017378006                     0.011451863            19
## 284:    31     3 0.005213402                     0.003082654            19
## 285:    32     1 0.001737801                     0.001371578            19
##         Seq
##      <char>
##   1:      U
##   2:      N
##   3:      N
##   4:      N
##   5:      N
##  ---
## 281:      V
## 282:      V
## 283:      N
## 284:      N
## 285:      N
```

```
## only show values within your window
EndDistribution <- EndDistribution[width %in% aRange]
```

```
##
## Plot heatmap of contexts
##


HeatmapPlot <- ggplot() + theme_bw() +
  geom_tile(data=EndDistribution,
            aes(x=width, y=ContextNumber, fill=Percent),
            color="white", size = 0.4) +
  geom_text(data=EndDistribution[Seq != "N"],
            aes(x=width, y= ContextNumber, label=Seq, size=4)) +
  scale_fill_gradient(low="white",high="maroon4",
                      limits=c(0, max(EndDistribution[["Percent"]])),
                      name = "% of 3'-ends") +
  xlab("Nucleotides downstream of 5'-end") +
  ylab("a piRNA nucleotide contexts") +
  scale_x_continuous(expand = c(0.000,0.0030)) +
  scale_size_continuous(guide="none") +
  scale_y_continuous(trans="reverse", breaks = NULL, expand = c(0,0)) +
  coord_equal() +
  theme(panel.background = element_blank(),
        panel.grid = element_blank(),
        legend.position = "top",
        axis.text.y=element_blank()); HeatmapPlot
```



```
##
## Plot Heatmap Deviation
##

HeatmapDeviationPlot <- ggplot() + theme_bw() +
  geom_tile(data=EndDistribution,
            aes(x=width, y=ContextNumber, fill=DeviationFromNormalDistribution),
            color="white", size = 0.4) +
  geom_text(data=EndDistribution[Seq != "N"],
            aes(x=width, y= ContextNumber, label=Seq, size=4))+
  scale_fill_gradient2(low="navy", mid="white", high="firebrick4",
                       limits = c(min(EndDistribution[["DeviationFromNormalDistribution"]]),
                                  max(EndDistribution[["DeviationFromNormalDistribution"]]))) +
  xlab("Nucleotides downstream of 5'-end") +
  ylab("a piRNA nucleotide contexts") +
  scale_x_continuous(expand = c(0.000,0.0030))+
  scale_size_continuous(guide= "none")+
  scale_y_continuous(trans="reverse", breaks = NULL, expand = c(0,0)) +
  coord_equal() +
  theme(panel.background = element_blank(),
        panel.grid = element_blank(),
        legend.position = "top",
        axis.text.y=element_blank()); HeatmapDeviationPlot
```

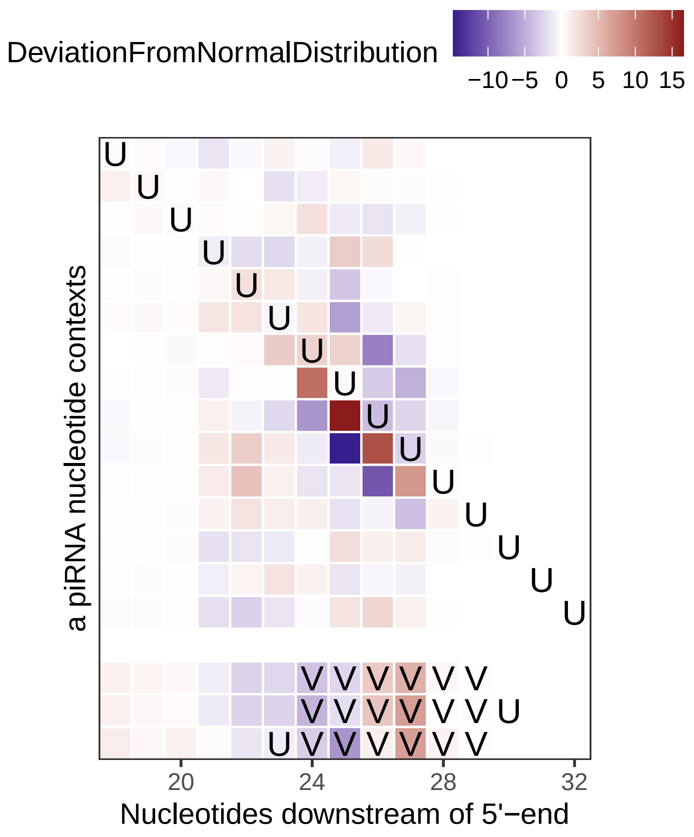


Figure 4 Z-score: Distribution of 3'-end relative to sequence context of interest

```r
## Crate main objects

GR <- original.GR
uniq.GR <- GR[mcols(GR)[["NH"]] %in% 1]

## Extend both ends of piRNAs to see sequence immediately down- and up-stream

ExpandBy <- 60
end(uniq.GR) <- end(uniq.GR)+ExpandBy
start(uniq.GR) <- start(uniq.GR)-ExpandBy

## Make sure that the extended sequences are still within the boundaries of chromosomes
## Use the appropriate genome
RefGen <- BSgenome::getBSgenome("BSgenome.Dmelanogaster.UCSC.dm6")
RefGen <- GRanges(seqnames = names(RefGen),
                  ranges = IRanges(start = 1,
                                   end = GenomeInfoDb::seqlengths(RefGen)))
uniq.GR <- IRanges::subsetByOverlaps(uniq.GR, RefGen, type = "within")

## Convert to data.table and obtain sequence

uniq.DT <- as.data.table(uniq.GR)
uniq.DT[["seq"]] <- as.vector(getSeq(BSgenome.Dmelanogaster.UCSC.dm6, uniq.GR))

## Select the range of analysis, must be less than what the original sequence were expanded by (60)
```

```r
Range  <- 51

## Select sequence context to analyze
## Add any custom contexts you would like to use
##    - use shorthands such as V = C, G, or A

Context <- "VVUVV"

## Make sequence context same length as the Range by adding N's
Context <- paste(c(paste(rep("N",floor((Range-nchar(Context))/2)), collapse=""), Context, paste(rep("N"

Context
```

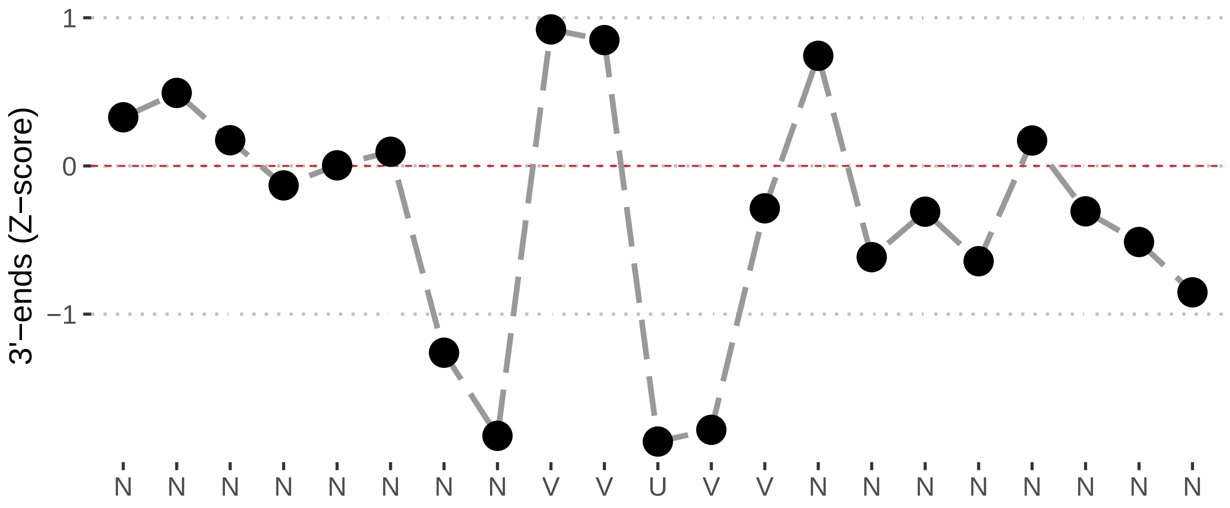
```
## [1] "NNNNNNNNNNNNNNNNNNNNNNNVVUVVNNNNNNNNNNNNNNNNNNNNNNN"
```

```r
## Convert Context to R regex format
Regex <- gsub("N", "[UCGA]", Context)
Regex <- gsub("V", "[CGA]", Regex)

## Convert to DNA
Regex <- gsub("U", "T", Regex)

## Create vector of positions for analysis
PosVector <- c(-((Range-1)/2):((Range-1)/2))

## Loop through the searching of the Context motif
## Use parallel package to speed up the Calculations
NumberOfCores <- 2

MotifSearch <- bind_rows(mclapply(seq_along(PosVector), function(a){

  Position <- PosVector[a]
  MaxWindow <- max(PosVector)

  ## Calculate total number of 3'-ends at (PosVector[a]) nucleotides away from center of context
  ThreePrimeSum <- sum(uniq.DT[str_detect(substring(seq,
                                          (width-ExpandBy)-(a+MaxWindow),
                                          (((width-ExpandBy)-(a+MaxWindow))+(Range-1))), Regex)][["MU

  ## prepare output
  Output <- data.table(
              Position = Position,
              Nucleotide = substring(Context, a, a),
              ThreePrime = ThreePrimeSum
              )

  return(Output)
  }, mc.cores = NumberOfCores))


MotifSearch[20:35]
```

```
##     Position Nucleotide ThreePrime
##        <int>     <char>      <int>
##  1:       -6          N     182832
##  2:       -5          N     185349
##  3:       -4          N     148123
##  4:       -3          N     132726
##  5:       -2          V     208009
##  6:       -1          V     206033
##  7:        0          U     131668
##  8:        1          V     133841
##  9:        2          V     174869
## 10:        3          N     203119
## 11:        4          N     165821
## 12:        5          N     174248
## 13:        6          N     165076
## 14:        7          N     187432
## 15:        8          N     174311
## 16:        9          N     168634
```

```
## Convert sum of 3'-ends at each position into a Z-score
MotifSearch[["Zscore"]] <- (MotifSearch[["ThreePrime"]] - mean(MotifSearch[["ThreePrime"]])) / sd(MotifS

## Only depict central 21 nucleotides
MotifSearchGraph <- MotifSearch[Position %in% c(-10:10)]
```

```
## Organize data
MotifSearchGraph[["Position"]] <- factor(MotifSearchGraph[["Position"]])

## Plot the Z-score graph

ZscorePlot <- ggplot() + theme_pubclean() +
  geom_hline(yintercept = 0, linetype="dashed", color = "red", lwd = 0.3) +
  geom_line(data=MotifSearchGraph,
            aes(x=Position, y=Zscore, group = 1),
            color = "gray60", size = 1, linetype = "longdash") +
  geom_point(data=MotifSearchGraph,
             aes(x=Position, y=Zscore),
             shape=16, size=5, fill = "gray60", color = "black") +
  labs(y= "3'-ends (Z-score)", x="")+
  scale_x_discrete(labels = MotifSearchGraph[["Nucleotide"]]) +
  theme(aspect.ratio = 0.4);ZscorePlot
```



-
-
-

This concludes the methods.

THE END.