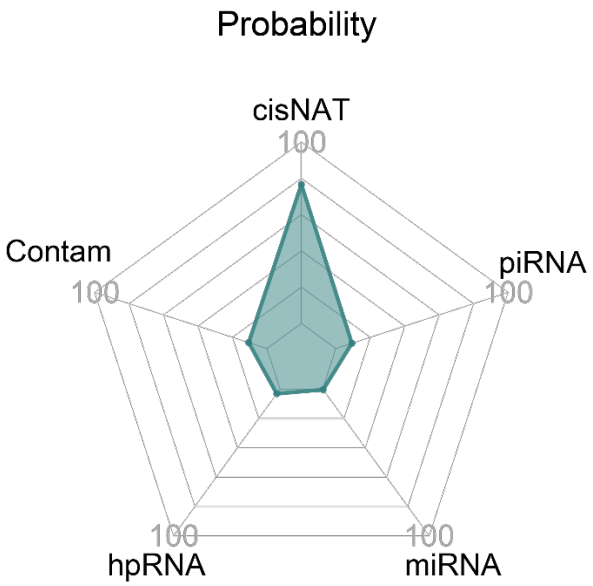
SUPPLEMENTARY MATERIAL

	Prediction	Validated class	Accuracy
False positive	Not contamination	Contamination	0
False negative	Contamination	Not contamination	10.96
True positive	Contamination	Contamination	15.07
True negative	Not contamination	Not contamination	73.97

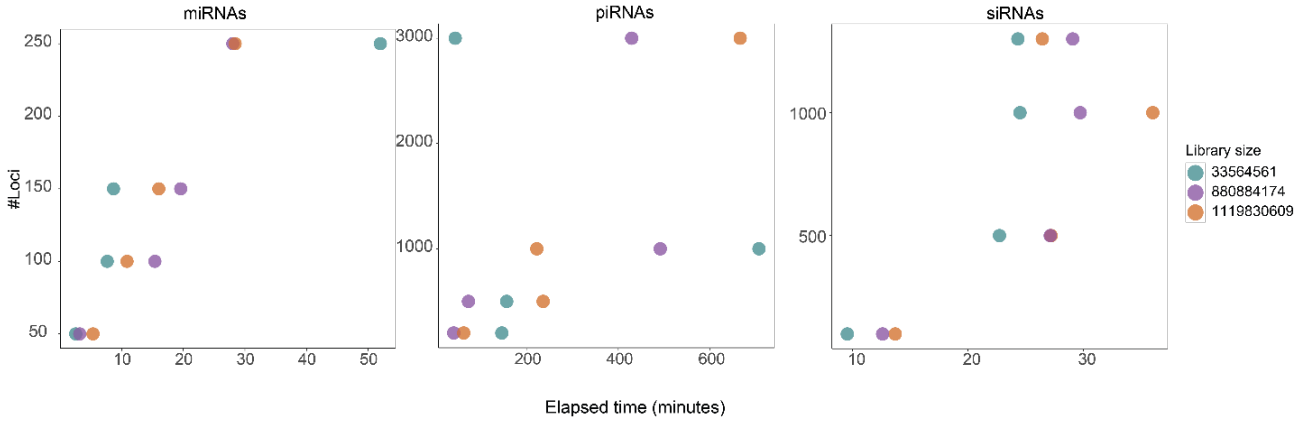
Supplemental Table 1. Accuracy of model-predicted classes versus real class for 74 loci withheld from all training further separated by true/false positives/negatives.

sRNA class	Accuracy
cisNAT	(12/14) 85.71%
hpRNA	(9/13) 69.23%*
miRNA	(9/11) 81.81%
piRNA	(24/24) 100%
contamination	(11/11) 100%

Supplemental Table 2. Model accuracy by sRNA class. Type I and II error rates of machine learning predictions by small RNA type for the 74 loci withheld from training.



Supplemental Figure 1. Probability (radar) plot of MiSiPi.RNA machine learning predictions. Radar plot made using the fmsb package (CRAN).



Supplemental Figure 2. Program run times (elapsed time) by number of loci and sequencing depth. Each module of the MiSiPi.RNA package was run on publicly available datasets of various library sizes. Elapsed time was recorded for each MiSiPi module with basic arguments in minutes using the system.time() function in R. SRA accessions used were: SRR22812151 (library size 33564561), SRR23418970 (library size 880884174), and SRR23418970 concatenated with SRR23418971 before mapping (library size 1119830609). miRNA and cisNAT siRNA annotations from *Drosophila melanogaster* GCF_000001215.4 were lifted to the *Drosophila simulans* PacBio genome (Chakraborty et al. 2021) using Liftoff 1.6.2.

Supplemental Methods

Accession number of libraries used in the study

melanogaster	simulans	spiderMite	planarian	zebrafish	mouse
SRR11680898	SRR22812151	SRR10513014	SRR12426750	SRR1554490	SRR13848060
SRR11680899	SRR22812152	SRR10513013	SRR12426752	SRR1554491	SRR13848061
SRR11680900	SRR22812153	SRR10513015	SRR12426754	SRR1554493	SRR13848062
SRR12313524	SRR22812154	SRR10513017	SRR12426756	SRR1554494	SRR24821465
SRR12313525	SRR22812155	SRR10513018	SRR12426758		SRR24821466
SRR12313526	SRR23418970	SRR10513019	SRR12426760		SRR24821469
	SRR23418971		SRR12426751		SRR24821470
			SRR12426753		
			SRR12426755		
			SRR12426757		
			SRR12426759		
			SRR12426761		

Method for identifying small RNA expressing regions of interest:

miRNAs & siRNAs: get reads of only miRNA and siRNA length

```
awk 'BEGIN {OFS = "\n"} {header = $0; getline seq; getline qheader ; getline
qseq ; if (length(seq) >= 19 && length(seq) <= 23) {print header, seq,
qheader, qseq}}' \
< trimmed.fq > small.fastq
```

piRNAs: get reads of piRNA length

```
awk 'BEGIN {OFS = "\n"} {header = $0; getline seq; getline qheader ; getline
qseq ; if(length(seq) >= 23 && length(seq) <= 30) {print header, seq, qheader,
qseq}}' \
< trimmed.fq > large.fastq
```

Realign reads to genome

```
# all small RNAs
awk 'BEGIN {OFS = "\n"} {header = $0; getline seq; getline qheader ; getline
qseq ; if (length(seq) >= 18 && length(seq) <= 32) {print header, seq,
qheader,  qseq}}' < trimmed.fq > all.fastq
```

```
#Align reads to genome
bowtie -p 10 -a -m 100 --best --strata --no-unal genome.fna all.fastq -S |
samtools view -@ 10 -q 10 -b |samtools sort -@ 10 -m 6G > all.bam
samtools index all.bam

bowtie -p 10 -a -m 100 --best --strata --no-unal genome.fna small.fastq -S
| samtools view -@ 10 -q 10 -b |samtools sort -@ 10 -m 6G > small.bam
samtools index small.bam

bowtie -p 20 -a -m 100 --no-unal genome.fna large.fastq -S | samtools view
-@ 10 -q 10 -b | samtools sort -@ 10 -m 6G > large.bam
samtools index large.bam
```

Calculate coverage of all reads over genome

```
bedtools genomecov -bg -ibam all.bam | awk '$4 > 100' > HE.tmp.bedgraph
```

Get large regions of high expression

```
bedtools merge -d 500 -i HE.tmp.bedgraph > HE.tmp.merge.bed

awk '{n=$2; x=$3; print $1"\t"$2"\t"$3"\t"x-n}' < HE.tmp.merge.bed |
awk '$4 > 40' > HE.all.bed #all highly expressed
```

Get potential regions of high miRNA/siRNA RNA expression

```
bedtools multicov -bams small.bam all.bam -bed HE.all.bed |
awk '{n=$5; x=$6; print $1"\t"$2"\t"$3"\t"n/x}' |
awk '$4 > 0.5' > HE.small.bed # only high expressed 21-23 nt
```

Get potential regions of high piRNA expression

```
bedtools multicov -bams large.bam all.bam -bed HE.all.bed |
awk '{n=$5; x=$6; print $1"\t"$2"\t"$3"\t"n/x}' | awk '$4 > 0.5' >
HE.large.bed #only high expressed 23-30 nt
```

Running the MiSiPi.RNA package:

Install MiSiPi.RNA. First install devtools and BiocManager

```
install.packages("devtools")

if (!require("BiocManager", quietly = TRUE))
  install.packages("BiocManager")

devtools::install_github("stupornova33/MiSiPi.RNA")
```

Running the MiSiPi.RNA on select loci (ROI) and alignments (bam file

```
library(MiSiPi.RNA)

vars <- set_vars(
    roi = "path/to/bed", # A bed file listing your regions of interest
    bam_file = "path/to/bam", # A BAM file of aligned reads. Index file must
also be present
    genome = "path/to/genome", # A genome fasta file. Chromosome names must
match the BED and BAM files
    plot_output = TRUE, # (TRUE or FALSE) If TRUE, MiSiPi.RNA will output
plots as pdfs
    path_to_RNAfold = "path/to/ViennaRNA/RNAfold", # Full path to RNAfold
executable
    path_to_RNAplot = "path/to/ViennaRNA/RNAplot", # Full path to RNAplot
executable
    pi_pal = "BlYel", # Palette option for the generated piRNA heatmap (see
below)
    si_pal = "RdYlBl", # Palette option for the generated siRNA heatmap (see
below)
    annotate_region = TRUE, # (TRUE or FALSE) Plots annotated gene features
below the hairpin arc plot which is useful for characterizing cisNAT loci
    weight_reads = "none", # Determines if read counts will be weighted.
("none", "locus_norm", "weight_by_prop", or an integer). This functionality
is in beta development.
    gtf_file = "path/to/gtf", # Full path to a 9 column GTF file. Required
    only if annotate_region is TRUE
             write_fastas = FALSE, # (TRUE or FALSE) If TRUE, MiSiPi.RNA will
write read pairs from functions to file. Default is FALSE
    out_type = "pdf" # ("pdf" or "png"),
    use_bed_names = F, # OPTIONAL (default is FALSE). Specifies whether
column 4 of the ROI file contains the locus name to use as file names
    density_timeout = 3600 # OPTIONAL (default is 3600s), for functions that
can run for a long time, move on to next locus after this amount of time

)

# Palettes:
# Palette options are:

# "RdYlBl", "BlYel", "yelOrRed", "MagYel", "Greens"


# Main Usage:
# choose one method or default is "all"
misipi_rna(vars, method = c("miRNA", "piRNA", "siRNA"))
```

Using ML module to evaluate loci identity

```
# After running with method = "all"
```

```
# To make the table for input to the machine learning model, run:
# The "_ml.txt" will be found in the misipi_output/date_timestamp/ directory.
ml_probability(path_to_table = "misipi_output/date_timestamp/", table =
"table_ml.txt")

# After any method, to create a sortable html summary with plots:
make_html_summary(path_to_tables = " misipi_output/date_timestamp/", type =
c("siRNA", "piRNA" or "miRNA"), ml_plots = FALSE) # if you didn't run the
ml_probability model
make_html_summary(path_to_tables = " misipi_output/date_timestamp/", type =
c("siRNA", "piRNA" or "miRNA"), ml_plots = TRUE) # if you ran the
ml_probability model

# Optional:
# For converting the .ps or .eps output files from the miRNA module to .png,
install ImageMagick and ghostscript, then run

ps2png(path_to_magick_exe, file_dir) or
eps2png(path_to_magick_exe, file_dir)
```

Changing the ML model for custom analysis

```
library(dplyr)

# All commented out code is optional and used for troubleshooting accuracy
or exploring your data.
# Read in output from misipi_rna(vars, method = "all") plus a column
containing the class and a column containing the total # of reads in the
bam file.
# This will be used for library size normalization of read counts.
# This file is in "run_all/" and will have the suffix "_ml.txt"
# and a column containing the library size of the BAM file used to make the
table
tbl <- read.table("path/to/machine/learning/output/your_data_ml.txt", sep =
"\t", header = TRUE)

# make sure rows are unique
tbl <- tbl %>% dplyr::distinct(log_shap_p,auc, strand_bias,perc_GC,
                               ave_size, perc_first_nucT, perc_A10,
highest_si_col, si_dicerz, num_si_dicer_reads,
                               hp_perc_paired, hp_phasedz, hp_mfe,
hp_dicerz, mi_perc_paired, mirna_dicerz, mirna_mfe,
                               mirna_overlapz, pingpong_col, max_pi_count,
max_piz_overlap, pi_phasedz, pi_phased26z,
                               unique_read_bias, type, species, libsize, X,
.keep_all = TRUE)

# Deal with missing max_piz_overlaps
idx <- which(is.na(tbl$max_piz_overlap))
tbl$max_piz_overlap[idx] <- -33
```

```
664
665     idx <- which(is.na(tbl$hp_mfe))
666     idx2 <- which(tbl$hp_mfe == 0)
667
668     #set missing hp MFE's to unlikely number
669     tbl$hp_mfe[idx] <- 10000
670     tbl$hp_mfe[idx2] <- 10000
671
672     idx <- which(tbl$mirna_mfe == 0)
673     tbl$mirna_mfe[idx] <- 1000
674
675     for(i in 1:nrow(tbl)){
676       if(tbl$max_pi_count[i] == -33){
677         tbl$max_pi_count[i] <- 0
678       }
679     }
680
681     # Group all types of contamination into one class
682     for(i in 1:nrow(tbl)){
683       if(tbl$type[i] == "snoRNA" || tbl$type[i] == "tRNA" || tbl$type[i] ==
684     "rRNA" || tbl$type[i] == "sno"){
685         tbl$type[i] <- "contamination"
686       }
687     }
688
689     # Normalize counts by libsize
690
691     tbl <- tbl %>% dplyr::mutate(pi_norm = (max_pi_count/libsize/1000000),
692     si_norm = (num_si_dicer_reads/libsize/1000000))
693
694     all_tbl <- tbl %>% dplyr::select(-c(locus_length, species, X, max_pi_count,
695     num_si_dicer_reads, libsize))
696     all_tbl$pi_norm <- as.numeric(all_tbl$pi_norm)
697     all_tbl$si_norm <- as.numeric(all_tbl$si_norm)
698
699     #write.table(all_tbl, file = "new_ML_contam_more_libsize_norm_mod.txt", sep
700     = "\t", quote = FALSE, row.names = FALSE, col.names = TRUE)
701     # install required packages if running for the first time
702     library(dplyr)   # for data cleaning
703     library(CatEncoders)
704     library(xgboost)
705     library(e1071)
706
707     # The MFE doesn't need to be negative, so take absolute value
708     all_tbl$mirna_mfe <- abs(all_tbl$mirna_mfe)
709
710     all_tbl$hp_mfe <- abs(all_tbl$hp_mfe)
711
712     # remove the locus name and
713     # Hot encode the type class
```

```
714     tbl <- all_tbl %>% dplyr::select(-c(locus))
715     train_labs <- LabelEncoder.fit(tbl$type)
716     tbl$type <- transform(train_labs, tbl$type)
717     # tranform to be zero-based
718     tbl$type <- tbl$type - 1
719
720     # check whether there is a class imbalance
721     table(tbl$type)                      # Raw class counts
722     prop.table(table(tbl$type)) * 100  #
723
724     # If serious class imbalance exists, consider methods to either re-sample,
725     # add more data, or use class weighting.
726     # tbl$type <- as.factor(tbl$type)  # caret requires a factor
727
728     # or can use SMOTE:
729     # Better than simple upsampling — creates synthetic examples of minority
730     classes.
731     # remotes::install_github("cran/DMwR")
732     #  library(DMwR)
733     #
734     # tbl$type <- as.factor(tbl$type)
735     # balanced_data <- SMOTE(type ~ ., data = tbl, perc.over = 100, perc.under
736     = 200)
737     # tbl <- balanced_data
738
739     # Create Training and Test data -
740     set.seed(100)  # setting seed to reproduce results of random sampling
741     # Segregate 80% of data for training and leave 20% out for cross validation
742     trainingRowIndex <- sample(1:nrow(tbl), 0.8*nrow(tbl))# row indices for
743     training data
744     trainingData <- tbl[trainingRowIndex, ] #%>% dplyr::select(-c(locus,
745     species))  # model training data
746     testData  <- tbl[-trainingRowIndex, ]  #%>% dplyr::select(-c(locus,
747     species))  # test data
748
749     train_x <- model.matrix(type ~ ., trainingData)[, -1]
750
751     test_x <- model.matrix(type ~ ., testData)[, -1]
752
753     train_y <- trainingData$type
754
755     test_y <- testData$type
756
757     train_y <- as.integer(as.character(train_y))  # safely convert factor to
758     numeric
759     test_y  <- as.integer(as.character(test_y))
760     # convert the train and test data into xgboost matrix type.
761     dtrain = xgb.DMatrix(data=train_x, label=train_y)
762     dval = xgb.DMatrix(data=test_x, label=test_y)
763
```

```
# Store X and Y for later use
#y = train_y
#x = train_x

options(scipen = 999)

#featurePlot(x = x,
#            y = as.factor(y),
#            plot = "density",
#            strip = strip.custom(par.strip.text = list(cex = .7)),
#            scales = list(x = list(relation="free"),
#                          y = list(relation="free")))

## read in the cross-validation table and transform the values same as the
training data
# I made my own cross-validation with completely unseen data to get a
better idea of real life accuracy,
# as the held-back training data seemed to have inflated accuracies.
# This table is formatted the same as the ml table.

Dat <-
read.table("C:/Users/tmjar/Desktop/MiSiPi_Stuff/machine_learning_species/ne
w_all_species/table_versions/all_species_w_whitefly_CV.txt", header = TRUE)

# Group all types of contamination as one class
dat <- dat %>% dplyr::select(-c(locus, locus_length, X,species))
for(i in 1:nrow(dat)){
  if(dat$type[i] == "snoRNA" || dat$type[i] == "tRNA" || dat$type[i] ==
"rRNA" || dat$type[i] == "sno" || dat$type[i] == "contamination"){
    dat$type[i] <- "contamination"
  }
}


dat <- dat %>% dplyr::mutate(pi_norm = "", si_norm = "")

#normalize counts by libsize

for(i in 1:nrow(dat)){
  #idx <- which(df$dataset == dat$X[i])
  dat$pi_norm[i] <- dat$max_pi_count[i]/dat$libsize[i]/1000000
  dat$si_norm[i] <- dat$num_si_dicer_reads[i]/dat$libsize[i]/1000000

}
dat$pi_norm <- as.numeric(dat$pi_norm)
dat$si_norm <- as.numeric(dat$si_norm)
dat <- dat %>% dplyr::select(-c(libsize, max_pi_count, num_si_dicer_reads))

test_labs <- LabelEncoder.fit(dat$type)
dat$type <- transform(test_labs, dat$type)
```

```
814     dat$type <- dat$type - 1
815
816     idx <- which(is.na(dat$max_piz_overlap))
817     dat$max_piz_overlap[idx] <- -33
818
819     # Set missing MFE to unlikely number
820
821     idx1 <- which(is.na(dat$hp_mfe))
822     idx2 <- which(dat$hp_mfe == 0)
823     dat$hp_mfe[idx1] <- 10000
824     dat$hp_mfe[idx2] <- 10000
825
826     idx1 <- which(is.na(dat$mirna_mfe))
827     idx2 <- which(dat$mirna_mfe == 0)
828
829     dat$mirna_mfe[idx1] <- 1000
830     dat$mirna_mfe[idx2] <- 1000
831
832     dat$hp_mfe <- abs(dat$hp_mfe)
833
834     new_data_y <- dat$type
835     #new_data_y <- new_data_y - 1
836     new_data_x <- dat %>% dplyr::select(-type)
837
838     ####### Do the training
839
840     rounds <- c(700,800,900,1000)
841     etas1 <- c(seq(0.01, 0.3, 0.05))
842     #etas <- c(etas1, etas2)
843     #etas <- c(seq(0.01, 0.4, 0.1))
844     subs <- c(0.7,0.75,0.8, 0.85,0.9, 0.95, 0.1)
845     depths <- c(5,6,7,8,9,10,11)
846     lambdas <- c(0.01,0.02,0.03,0.04,0.05,0.06,0.07,0.08,0.09,0.1,1,0)
847     alphas <- c(0,0.1,0.5,1.0)
848     gammas <- c(0.05,0.06,0.07,0.08,0.09,0.1,0.3,0.5,0.7,0.9,1.0)
849
850     len <- length(gammas)
851     n_iter <- len
852
853     pb <- txtProgressBar(min = 0,
854                          max = n_iter,
855                          style = 3,
856                          width = 50,
857                          char = "=")
858
859     results <- data.frame(par = numeric(length = length(len)), mean_accuracy =
860     numeric(length = length(len)))
861
862     for( i in 1:length(gammas)){
863       set.seed(1234)
```

```
  #run with softmax for quick accuracy check, pick best model, re-run with softprob and save
  cur_model <- xgboost(data = dtrain, max.depth = 6, subsample = 0.85, eta = 0.01, nthread = 2, lambda = 0.01, alpha = 0.5,
                       gamma = 0.05, eval_metric = "mlogloss", nrounds = 700, objective = "multi:softmax", num_class = 5, verbose = 0,
                       early_stopping_rounds = 10, print_every_n = 10)

  # once final model is selected, run the following to get a model that outputs percent probability instead of just "class"
  # prob_model <- xgboost(data = dtrain, max.depth = 6, eval_metric = "mlogloss", eta = 0.01, subsample = 0.85, nthread = 2,
  #                       alpha = 0.1, lambda = 0.01, gamma = 0.05,
  #                       nrounds = 700, objective = "multi:softprob", num_class = 5, verbose = FALSE)

  # then save prob_model to a .rds file
  real_pred <- predict(cur_model, as.matrix(new_data_x))
  #df <- as.data.frame(matrix(real_pred, ncol = 2, byrow = TRUE)) #%>% dplyr::select(c(V1))

  prob_mat <- matrix(real_pred, ncol = 5, nrow = nrow(new_data_x))
  cur_err <- mean(real_pred != new_data_y)
  cur_accuracy <- 1 - cur_err
  print(cur_accuracy)

  results[i,] <- c(gammas[i], cur_accuracy)
  setTxtProgressBar(pb, i)
}

# To check whether model is overfit
# log <- model$evaluation_log
# plot(log$iter, log$train_mlogloss, type = "l", col = "blue")
# plot(log$iter, log$eval_mlogloss, type = "l", col = "red")
# legend("topright", legend = c("Train", "Validation"), col = c("blue", "red"), lty = 1)

# save the model
xgb.save(prob_model, "path/to/xgb_tuned.rds")

# load the model
all_model <- xgb.load("path/to/xgb_tuned.rds")

# Calculate Type I and Type II error rates
all_pred <- predict(all_model, as.matrix(new_data_x))
all_df <- as.data.frame(matrix(all_pred, ncol = 5, byrow = TRUE))
#all_pred <- all_pred[,1]
colnames(all_df) <- c("prob_cis", "prob_contam", "prob_hp", "prob_mi", "prob_pi")
all_df$real_type <- dat$type
```

```
914
915     for(i in 1:nrow(all_df)){
916       real <- all_df$real_type[i]
917       #true and contam
918       #true and not contam
919       #false and contam
920       #false and not contam
921       max <- which(all_df[i,1:5] == max(all_df[i,1:5]))
922       if((max - 1) == real){
923         all_df$cond[i] <- "TRUE"
924       } else {
925         all_df$cond[i] <- "FALSE"
926       }
927
928       if(all_df$real_type[i] == 1){
929         all_df$contam[i] <- "contam"
930       } else {
931         all_df$contam[i] <- "not_contam"
932       }
933     }
934
```