Source Data Code Documentation

This vignette describes the computational methods associated with the manuscript. The code is available on GitHub (https://github.com/HaaseLab).

Functions used in the notebooks

```
# buildTranscriptomes.R

# Functions needed for the buildTranscriptomes.R script:
# GenomicRanges, txdbmaker, Rsamtools, BSgenome, dplyr, IRanges, Biostrings and the
BSgenome package for the used genome

# Function to create sequences with gene information and 2000 N (or custom buffer
length) mask between genes
create_sequences <- function(gr, genome, nameCol = "gene_name", output_dir,
buffer_length = 2000) {
    start_time <- Sys.time()
    message("Function 'create_sequences' started at: ", start_time)

    if (!(nameCol %in% colnames(mcols(gr)))) {
        stop("nameCol '", nameCol, "' not found in metadata; available: ",
         paste(colnames(mcols(gr)), collapse = ", "))
    }

    buffer <- strrep("N", buffer_length)

    # Group by chromosome and gene_name, then sort
    gr_by_chr <- split(gr, seqnames(gr))
    gr_by_chr <- lapply(gr_by_chr, function(chr_gr) {
        chr_gr_by_gene <- split(chr_gr, mcols(chr_gr)[[nameCol]])
        chr_gr_by_gene <- chr_gr_by_gene[order(sapply(chr_gr_by_gene, function(x)
start(x)[1]))]
        return(chr_gr_by_gene)
    })

    sequences <- list()
    for (chr in names(gr_by_chr)) {
        chr_seq <- buffer  # Start each chromosome with buffer
        for (gene in names(gr_by_chr[[chr]])) {
            gene_ranges <- gr_by_chr[[chr]][[gene]]
            seq <- ""
            for (i in seq_along(gene_ranges)) {
                range <- gene_ranges[i]
                coord <- paste0(as.character(seqnames(range)), ":", start(range), "-",
end(range))
                seq <- paste0(seq, as.character(getSeq(genome, GRanges(coord))))
            }
            chr_seq <- paste0(chr_seq, seq, buffer)
            sequences[[gene]] <- list(gene_name = gene, sequence = seq, original_coords
= gene_ranges)
        }
    sequences[[paste0("chr_", chr)]] <- chr_seq
    }
    mid_time <- Sys.time()
```

```
    message("Function 'create_sequences' (without saving it as FASTA) at: ", mid_time)
    message("Execution time without saving it as FASTA: ",
            difftime(mid_time, start_time, units = "auto"))

    # Get chromosome entries
    chr_sequences <- sequences[grep("^chr_", names(sequences))]
    chr_sequences <- lapply(chr_sequences, function(seq) {
        names(seq) <- sub("^chr_", "$chr_", names(seq))
        seq
    })

    # Convert to DNAStringSet, set names and write to FASTA file
    fasta_sequences <- DNAStringSet(unlist(chr_sequences))
    names(fasta_sequences) <- sub("chr_", "", names(chr_sequences))
    fasta_file <- paste0(output_dir, "piCtranscriptome.fasta")
    writeXStringSet(fasta_sequences, fasta_file)
    indexFa(fasta_file)
    message("FASTA file has been saved as ", fasta_file)

    # Print some information
    message("Number of genes processed: ", length(fasta_sequences) -
length(grep("^chr_", names(fasta_sequences))))
    message("Number of chromosomes represented: ", length(grep("^chr_",
names(fasta_sequences))))
    message("Total sequence length: ", sum(width(fasta_sequences)))

    end_time <- Sys.time()
    message("Function 'create_sequences' ended at: ", end_time)
    message("Total execution time: ", difftime(end_time, start_time, units = "auto"))

    return(sequences)
}


# Create GTF
create_gtf <- function(gr, sequences, nameCol = "gene_name", output_dir, buffer_length =
2000) {
    start_time <- Sys.time()
    message("Function 'create_gtf' started at: ", start_time)

    gtf_data <- data.frame()

    # Reconstruct gr_by_chr to get the genes in order(first by chr, then by start
position)
    gr_by_chr <- split(gr, seqnames(gr))
    gr_by_chr <- lapply(gr_by_chr, function(chr_gr) {
        chr_gr_by_gene <- split(chr_gr, mcols(chr_gr)[[nameCol]])
        chr_gr_by_gene <- chr_gr_by_gene[order(sapply(chr_gr_by_gene, function(x)
start(x)[1]))]
        return(chr_gr_by_gene)
    })

    for (chr in names(gr_by_chr)) {
        chr_name <- chr
        current_pos <- buffer_length + 1  # Start position in the custom transcriptome
for this chromosome

        chr_genes <- gr_by_chr[[chr_name]]

        for (gene in names(chr_genes)) {
```

```
            gene_info <- sequences[[gene]]
            gene_ranges <- gene_info$original_coords

            gene_length <- nchar(gene_info$sequence)
            gene_start <- current_pos  # Start after the initial buffer
            gene_end <- gene_start + gene_length - 1

            # Gene entry
            gtf_data <- rbind(gtf_data, data.frame(
                seqname = chr_name,
                source = "custom",
                feature = "gene",
                start = gene_start,
                end = gene_end,
                score = ".",
                strand = as.character(strand(gene_ranges[1])),
                frame = ".",
                attribute = paste0('gene_id "', gene, '"; ',
                                   'gene_name "', gene, '"; ',
                                   'transcriptome_start "', gene_start, '"; ',
                                   'transcriptome_end "', gene_end, '"; ',
                                   'original_chr "',
as.character(seqnames(gene_ranges[1])), '"; ',
                                   'original_start "', min(start(gene_ranges)), '"; ',
                                   'original_end "', max(end(gene_ranges)), '";')
            ))

            # Feature entries (by type: CDS, 5UTR, 3UTR) or just exon (default)
            exon_cumulative_start <- gene_start
            for (i in seq_along(gene_ranges)) {
                range <- gene_ranges[i]

                range_length <- width(range)
                feature_start <- exon_cumulative_start
                feature_end <- exon_cumulative_start + range_length - 1

                # Use "exon" as default feature type since type column is missing
                feature_type <- if("type" %in% colnames(mcols(range))) {
                    as.character(range$type)
                } else {
                    "exon"
                }

                gtf_data <- rbind(gtf_data, data.frame(
                seqname = chr_name,
                source = "custom",
                feature = feature_type,
                start = feature_start,
                end = feature_end,
                score = ".",
                strand = as.character(strand(range)),
                frame = ".",
                attribute = paste0('gene_id "', gene, '"; ',
                                   'gene_name "', gene, '"; ',
                                   'transcript_id "', gene, '_transcript"; ',
                                   'original_chr "', as.character(seqnames(range)), '";
',
                                   'original_start "', start(range), '"; ',
                                   'original_end "', end(range), '"; ',
                                   'original_strand "', strand(range), '";')
```

```
                ))

                # Update the cumulative start position
                exon_cumulative_start <- feature_end + 1
            }

            current_pos <- gene_end + buffer_length + 1  # Add buffer length for the
next gene
        }
    }
    mid_time <- Sys.time()
    message("Function 'create_gtf' (without saving it as GTF) at: ", mid_time)
    message("Execution time without saving it as GTF: ",
            difftime(mid_time, start_time, units = "auto"))

    # Save GTF file
    gtf_file <- paste0(output_dir, "piCtranscriptome.gtf")
    rtracklayer::export(gtf_data, gtf_file, format = "gtf")
    message("GTF file has been saved as", gtf_file)

    end_time <- Sys.time()
    message("Function 'create_gtf' ended at: ", end_time)
    message("Total execution time: ",
            difftime(end_time, start_time, units = "auto"))

    return(gtf_data)
}

get_piC_transcriptome <- function(piCs_gr, genome, column_name_to_sort_by, output_dir,
buffer_length = 2000) {
    # Sort the GRanges object
    sorted_piCs_gr <- sort(piCs_gr)

    # Access the metadata columns
    metadata_columns <- mcols(sorted_piCs_gr)

    # Check if the column exists
    if (!column_name_to_sort_by %in% colnames(metadata_columns)) {
        stop("Column", column_name_to_sort_by, "not found in metadata.")
    }

    # Sort by the specified column using variable name
    sort_order <- order(metadata_columns[[column_name_to_sort_by]])
    sorted_piCs_gr <- sorted_piCs_gr[sort_order]

    # Get the sorted metadata for creating gene names
    sorted_metadata <- mcols(sorted_piCs_gr)

    # Create a new column that is the rank of the piC
    mcols(sorted_piCs_gr)$gene_name <- paste0("rank_",
sorted_metadata[[column_name_to_sort_by]])

    # Create sequences
    message(" ** Creating sequences with buffer length of ", buffer_length, " Ns **")
    sequences <- create_sequences(sorted_piCs_gr, genome, nameCol, output_dir,
buffer_length = buffer_length)

    # Check if sequences is empty or NULL
    if (is.null(sequences) || length(sequences) == 0) {
        stop("Error: No sequences were created. Please check the input data and the
```

```
create_sequences function.")
    }

    # Create and write GTF file
    message(" ** Creating GTF file **")
    gtf_data <- create_gtf(sorted_piCs_gr, sequences, output_dir, buffer_length =
buffer_length)


    message(" ** Transcriptome has been created and saved in ", output_dir, " **")
}

getGeneTranscriptome <- function(genes, genome, nameCol = "gene_name", output_dir,
buffer_length = 2000, includes = "PCGonly") {
    # genes
    if (class(genes) == "GRanges") {
        genes_gr <- genes
    } else if (class(genes) == "character") {
        message("Loading gtf-file from provided directory.")
        genes_gr <- rtracklayer::import(genes)
    } else {
        stop("Error: genes is not a GRanges object or gtf-directory.")
    }

    #filter genes by 'includes' (all genes or just protein-coding genes)
    if (includes == "PCGonly") { # PCG: protein-coding genes only
        message("Only protein-coding genes will be included in the transcriptome")
        # Filter by protein-coding genes
        genes_gr<-genes_gr[genes_gr$gene_id %in% unique(genes_gr[genes_gr$type %in%
"CDS"]$gene_id)]
        genes_gr<-genes_gr[genes_gr$transcript_id %in% unique(genes_gr[genes_gr$type
%in% "CDS"]$transcript_id)] #since some transcripts do not contain all PCG annotation
features (CDS etc)

        #remove predicted subset (XM_* or XR_*)
        genes_gr <- genes_gr[!grepl("^XM_|^XR_", genes_gr$transcript_id)]
        message("Predicted transcripts (XM_* or XR_*) removed.")

        message("Number of remaining genes: ",
                length(as.list(unique(genes_gr$gene_id))))
        message("Number of remaining transcripts: ",
                length(as.list(unique(genes_gr$transcript_id))))
    } else if (includes == "totalGenes_PCG_nonPCG") {
        message("All genes (PCG and non-PCG) will be included in the transcriptome")
        message("Total genes and total transcripts: ",
length(as.list(unique(genes_gr$gene_id))), " and ",
length(as.list(unique(genes_gr$transcript_id))))
    } else {
        stop("Invalid option for includes. Please choose 'PCGonly' or
'totalGenes_PCG_nonPCG'")
    }

    # Assign stop_codon to CDS
    genes_gr$type[genes_gr$type == "stop_codon"] <- "CDS"
    # check if UTRs are annotated in GRanges, if not create them
    if (all(!c("3UTR", "5UTR") %in% unique(genes_gr$type))) {
        if (class(genes) == "character") {
            message("UTR annotations not present. Trying to load gtf with
makeTxDbFromGFF() if 'genes' variable was a directory")
            genes_txdb <- makeTxDbFromGFF(genes)
```

```
            transcript_to_gene_name <- setNames(mcols(genes_gr)[[nameCol]],
genes_gr$transcript_id)
            transcript_to_gene <- setNames(mcols(genes_gr)[[nameCol]],
genes_gr$transcript_id)
            transcript_to_gene <- transcript_to_gene[!is.na(names(transcript_to_gene))]
            # get 5'UTR
            fiveUTRs_gr <- unlist(fiveUTRsByTranscript(genes_txdb, use.names = TRUE),
use.names = TRUE)
            fiveUTRs_gr$transcript_id <- names(fiveUTRs_gr)
            fiveUTRs_gr$gene_id <- transcript_to_gene[fiveUTRs_gr$transcript_id]
            fiveUTRs_gr$gene_name <- transcript_to_gene_name[fiveUTRs_gr$transcript_id]
            fiveUTRs_gr$type <- "5UTR"
            names(fiveUTRs_gr) <- NULL

            # get 3'UTR
            threeUTRs_gr <- unlist(threeUTRsByTranscript(genes_txdb, use.names = TRUE),
use.names = TRUE)
            threeUTRs_gr$transcript_id <- names(threeUTRs_gr)
            threeUTRs_gr$gene_id <- transcript_to_gene[threeUTRs_gr$transcript_id]
            threeUTRs_gr$gene_name <-
transcript_to_gene_name[threeUTRs_gr$transcript_id]
            threeUTRs_gr$type <- "3UTR"
            names(threeUTRs_gr) <- NULL

            genes_gr <- c(genes_gr, fiveUTRs_gr, threeUTRs_gr)
            genes_gr <- sort(genes_gr)
            #save GRanges with UTRs
            gtf_newFile <- sub("\\.[^./]*$", "", sub(".*/", "", genes))
            gtf_file <- paste0(output_dir, gtf_newFile, "wUTRs.gtf")
            rtracklayer::export(genes_gr, gtf_file, format = "gtf")
            message("GTF file with UTRs has been saved as ", gtf_file)
        } else {
            stop("Error: UTRs not present in GRanges object, either include them or run
this function with the directory of your gtf-file")
        }
    }

    # Create a list to store the reduced GRanges for each gene
    reduced_genes_list <- list()

    # Sort the GRanges object
    genes_gr <- sort(genes_gr)

    # Get unique gene_ids/gene_names
    unique_genes <- unique(mcols(genes_gr)[[nameCol]])

    # Collapse each gene with annotation hierarchy CDS > 3´UTR > 5´UTR
    for (gene in unique_genes) {
        # Subset data for the current gene
        gene_data <- genes_gr[mcols(genes_gr)[[nameCol]] == gene]

        # Reduce all regions first
        cds_reduced <- reduce(gene_data[gene_data$type == "CDS"])
        utr5_reduced <- reduce(gene_data[gene_data$type == "5UTR"])
        utr3_reduced <- reduce(gene_data[gene_data$type == "3UTR"])

        # Remove CDS regions from UTRs
        if (length(cds_reduced) > 0) {
            utr5_reduced <- GenomicRanges::setdiff(utr5_reduced, cds_reduced)
            utr3_reduced <- GenomicRanges::setdiff(utr3_reduced, cds_reduced)
```

```
        }

        # Handle overlapping 5'UTR and 3'UTR regions
        overlap <- GenomicRanges::intersect(utr5_reduced, utr3_reduced)
        if (length(overlap) > 0) {
            utr5_reduced <- GenomicRanges::setdiff(utr5_reduced, overlap)
            utr3_reduced <- GenomicRanges::union(utr3_reduced, overlap)
        }

        # Combine all regions
        gene_regions <- c(cds_reduced, utr5_reduced, utr3_reduced)

        # Sort the combined regions
        gene_regions <- sort(gene_regions)

        # Remove duplicates
        gene_regions <- unique(gene_regions)

        # Assign region types
        region_types <- rep("", length(gene_regions))
        region_types[gene_regions %over% cds_reduced] <- "CDS"
        region_types[gene_regions %over% utr5_reduced] <- "5UTR"
        region_types[gene_regions %over% utr3_reduced] <- "3UTR"

        # Add metadata
        mcols(gene_regions)$type <- region_types
        mcols(gene_regions)[[nameCol]] <- rep(gene, length(gene_regions))


        # Add to the list if there are any regions
        if (length(gene_regions) > 0) {
            reduced_genes_list[[gene]] <- gene_regions
        }
    }

    # Combine all reduced GRanges into a single GRanges object
    reduced_genes <- unlist(GRangesList(reduced_genes_list))

    # Remove any potential duplicates in the final GRanges object
    reduced_genes <- unique(reduced_genes)

    # Sort the final GRanges object
    reduced_genes <- sort(reduced_genes)
    if (nameCol == "gene_name") {
        reduced_genes$gene_id <- mcols(reduced_genes)[[nameCol]]
    } else if (nameCol == "gene_id") {
        reduced_genes$gene_name <- mcols(reduced_genes)[[nameCol]]
    }

    message("Number of ranges in the reduced GRanges: ", length(reduced_genes))
    message("Number of unique genes in the reduced GRanges: ",
length(unique(reduced_genes$gene_id)))

    # Save the reduced_genes object to an RData file
    save(reduced_genes, file = paste0(output_dir,
"collapsed_prioritizedCDS3UTR5UTR.RData"))

    # Write the GTF file
    gtf_file <- paste0(output_dir, "collapsed_prioritizedCDS3UTR5UTR.gtf")
    rtracklayer::export(reduced_genes, gtf_file, format = "gtf")
```

```
    message("Collapsed, prioritized gene file has been saved as ", gtf_file)

    # Create sequences
    message(" ** Creating sequences with buffer length of ", buffer_length, " Ns **")
    sequences <- create_sequences(reduced_genes, genome, nameCol = nameCol, output_dir,
buffer_length = buffer_length)

    # Check if sequences is empty or NULL
    if (is.null(sequences) || length(sequences) == 0) {
        stop("Error: No sequences were created. Please check the input data and the
create_sequences function.")
    }

    # Create and write GTF file
    message(" ** Creating GTF file **")
    gtf_data <- create_gtf(reduced_genes, sequences, nameCol = nameCol, output_dir,
buffer_length = buffer_length)


    message(" ** Transcriptome has been created and saved in ", output_dir, " **")


}
```

```
# plotLogo.R

# Logo from positions 1 to 15 and optional 3' end extension
logoPlot <- function(myAlignments_subset, sampleName, ylim = c(0, 2), genome = NULL){
    # Validate input
    if (!inherits(myAlignments_subset, "GRanges")) {
        stop("myAlignments_subset needs to be a GRanges object (e.g. select for primary
or unique piRNAs).")
    }
    if (!"seq" %in% colnames(mcols(myAlignments_subset))) {
        stop("myAlignments_subset needs to have a seq column. Get through PICBload with
parameter GET.ORIGINAL.SEQUENCE = TRUE.")
    }

    # Main 5' logo (positions 1..15, or fewer if shorter)
    main_width <- min(length(myAlignments_subset$seq)-2, 15)
    label_pos <- c(1, 5, 10, 15)
    main_seqs <- chartr('ATGC', 'AUGC', substr(as.character(myAlignments_subset$seq),
start = 1, stop = main_width))
    p_logo <- ggseqlogo(main_seqs, seq_type = 'rna') +
        scale_y_continuous(limits = ylim, breaks = seq(0, 2, 0.5)) +
        ggtitle(sampleName) +
        scale_x_continuous(breaks = label_pos, labels = label_pos[label_pos <=
main_width]) +
        theme(axis.ticks.y = element_line(),
            axis.ticks.x = element_line(),
            axis.line.x = element_line(),
            axis.line.y = element_line())

    # Attempt to build 3' extension logo if a BSgenome is provided/available
    ext_logo <- NULL
    if (!is.null(genome)) {
        # Resolve BSgenome object
        bs_obj <- NULL
        if (inherits(genome, "BSgenome")) {
```

```
            print("Using provided BSgenome object to build 3' extension logo.")
            bs_obj <- genome
        } else if (is.character(genome) && length(genome) == 1) {
            bs_obj <- tryCatch(BSgenome::getBSgenome(genome), error = function(e) NULL)
            print("Using provided BSgenome name to build 3' extension logo.")
        } else {
            stop("Invalid BSgenome object or name provided. Make sure to load the
BSgenome package beforehand or place it in quotes.")
        }

        if (!is.null(bs_obj)) {
            # Compute windows around the 3' end per read
            # + strand: 3' end = end(); take [-2,-1] and [+1,+2,+3]
            # - strand: 3' end = start(); take [-2,-1] -> [start+1, start+2] and
[+1..+3] -> [start-3, start-1]
            three_prime_regions <- IRanges(
                start = ifelse(strand(myAlignments_subset) == "+",
                               end(myAlignments_subset) - 2,
                               start(myAlignments_subset) - 2),
                end   = ifelse(strand(myAlignments_subset) == "+",
                               end(myAlignments_subset) + 2,
                               start(myAlignments_subset) + 2)
            )

            # Get the sequences
            three_prime_seqs <- BSgenome::getSeq(
                BSgenome.Mmusculus.UCSC.mm10,
                GRanges(seqnames(myAlignments_subset),
                    IRanges(three_prime_regions),
                    strand(myAlignments_subset)
                )
            )

            three_prime_seqs <- chartr('ATGC', 'AUGC',
substr(as.character(three_prime_seqs), start = 1, stop = 5))
            if (length(three_prime_seqs) > 0) {
                ext_logo <- ggseqlogo(three_prime_seqs, seq_type = 'rna') +
                    scale_y_continuous(limits = ylim, breaks = NULL) +
                    # Separator line at the left edge of the extension logo
                    scale_x_continuous(breaks = 3, labels = "3'end") +
                    theme(axis.title.y = element_blank(),
                        axis.text.y = element_blank(),
                        axis.ticks.y = element_blank(),
                        axis.ticks.x = element_line(),
                        axis.line.x = element_line(),
                        axis.line.y = element_line())

            }
        }
    }

    # Return either the base logo or the combined logo with extension
    if (!is.null(ext_logo)) {
        # Place extension to the right; relative widths reflect number of positions
        return(p_logo + ext_logo + patchwork::plot_layout(widths = c(main_width, 5)))
    } else {
        return(p_logo)
    }
}
```

```
# plotCoverage.R

# Comprehensive function to plot coverage tracks for GRanges of alignments (e.g. loaded
with PICBload), piRNA clusters and GTF files
coverageTracks <- function(fgr, fgr2, piRNAs_from_Bam=c(), chromosome, IRangesCoord,
tilesWidth=100, xWidth=10000, yRange=c(), normalize = "rpm", scaleWidthKB =
ifelse(end(IRangesCoord) - start(IRangesCoord) > 1000, 1, 0.2)) {
    # find total number of reads in library
    if ("MULT" %in% names(mcols(piRNAs_from_Bam))) {
        Total_reads<-sum(piRNAs_from_Bam$MULT)
    } else {
        Total_reads<-length(piRNAs_from_Bam)
    }


    # Make tiles (100nt default) from GRanges
    fgr.tiles<-unlist(tile(x = fgr, width = tilesWidth)) #minus strand
    fgr.tiles2<-unlist(tile(x = fgr2, width = tilesWidth)) #plus strand

    # Limit GRanges to selected coordinates
    piRNAs_selected_minus<-subsetByOverlaps(piRNAs_from_Bam, fgr, type = "within")
#minus strand
    piRNAs_selected_plus<-subsetByOverlaps(piRNAs_from_Bam, fgr2, type = "within") #plus
strand

    # For minus strand: find reads coverage (rcov) (using the uncollapsed GRanges)
    if ("MULT" %in% names(mcols(piRNAs_selected_minus))) {
        GRcov<-coverage(piRNAs_selected_minus, weight = piRNAs_selected_minus$MULT)
#minus strand
    } else {
        GRcov<-coverage(piRNAs_selected_minus) #minus strand
    }

    # For plus strand: find reads coverage (rcov) (using the uncollapsed GRanges)
    if ("MULT" %in% names(mcols(piRNAs_selected_plus))) {
        GRcov2<-coverage(piRNAs_selected_plus, weight = piRNAs_selected_plus$MULT) #plus
strand
    } else {
        GRcov2<-coverage(piRNAs_selected_plus) #plus strand
    }

    # Ensure coverage objects have same seqlevels as tiles (problem if you hand-currate
some regions beforehand)
    GRcov_filtered <- GRcov[seqlevels(fgr.tiles)] #minus strand
    GRcov2_filtered <- GRcov2[seqlevels(fgr.tiles2)] #plus strand

    # average the coverage in each tile (fgr.tiles) (ie. average coverage within each
tile) and normalize to the size of the library (rpm) -> new column under the name "ncov"
    if (normalize == "rpm") {
        message("Normalizing to RPM")
        fgr.tiles2.ncov<-binnedAverage(bins = fgr.tiles, numvar =
(GRcov_filtered*1000000)/Total_reads, varname = "ncov") #minus strand
        fgr.tiles2.ncov2<-binnedAverage(bins = fgr.tiles2, numvar =
(GRcov2_filtered*1000000)/Total_reads, varname = "ncov") #plus strand
    } else if (normalize == "region") {
        message("Normalizing to region")
        # devides by the total number of reads in the region (minus and plus strand
combined). Sum of ncov is therefore the average piRNA length since the coverage
considers each piRNA for its entire length
```

```
        fgr.tiles2.ncov<-binnedAverage(bins = fgr.tiles, numvar =
GRcov_filtered/length(c(piRNAs_selected_minus, piRNAs_selected_plus))*tilesWidth,
varname = "ncov") #minus strand
        fgr.tiles2.ncov2<-binnedAverage(bins = fgr.tiles2, numvar =
GRcov2_filtered/length(c(piRNAs_selected_minus, piRNAs_selected_plus))*tilesWidth,
varname = "ncov") #plus strand
    } else if (normalize == "none") {
        message("Not normalizing")
        fgr.tiles2.ncov<-binnedAverage(bins = fgr.tiles, numvar = GRcov_filtered,
varname = "ncov") #minus strand
        fgr.tiles2.ncov2<-binnedAverage(bins = fgr.tiles2, numvar = GRcov2_filtered,
varname = "ncov") #plus strand
    }

    ######## Minus strand ########
    fgr.tiles.ncov3<-as.data.frame(fgr.tiles2.ncov)
    fgr.tiles.ncov3$ncov<--fgr.tiles.ncov3$ncov # make minus strand negative

    ######## Plus strand ########
    fgr.tiles.ncov3_2<-as.data.frame(fgr.tiles2.ncov2)

    axisLinesDist <- max(fgr.tiles.ncov3$end) - min(fgr.tiles.ncov3$start) / 8

    yAxisPrep <- max(abs(min(fgr.tiles.ncov3$ncov)), abs(max(fgr.tiles.ncov3_2$ncov)),
1)

    # Determine rounding for y-axis breaks
    roundBy <- 0
    if (yAxisPrep <= 1) {
        yAxisBreaks <- 0.1
        roundBy <- 2
    } else if (yAxisPrep <= 10) {
        yAxisBreaks <- 1
        roundBy <- 0
    } else if (yAxisPrep <= 100) {
        yAxisBreaks <- 10
        roundBy <- -1
    } else if (yAxisPrep <= 1000) {
        yAxisBreaks <- 100
        roundBy <- -2
    } else if (yAxisPrep <= 10000) {
        yAxisBreaks <- 1000
        roundBy <- -3
    } else if (yAxisPrep > 10000) {
        yAxisBreaks <- 10000
        roundBy <- -4
    }

    ######## Coverage Track ########
    pltCvrg <- ggplot() + theme_classic() +
                geom_line(data = fgr.tiles.ncov3, aes(x = start, y = ncov),
color="blue") +
                #geom_area(data = fgr.tiles.ncov3, aes(x = start, y = ncov),
color="navyblue", fill="blue") + # can be uncommented to fill the area under the curve
                geom_line(data = fgr.tiles.ncov3_2, aes(x = start, y = ncov),
color="red") +
                #geom_area(data = fgr.tiles.ncov3_2, aes(x = start, y = ncov),
color="tomato4", fill="tomato1") + # can be uncommented to fill the area under the curve
                scale_x_continuous(breaks = seq(min(fgr.tiles.ncov3$start),
max(fgr.tiles.ncov3$end), by = xWidth)) +
```

```
                theme(aspect.ratio = 0.4,
                        axis.line.y = element_blank(),
                        panel.grid.minor = element_blank()) +
                coord_cartesian(xlim = c(start(IRangesCoord), end(IRangesCoord))) + #,
ylim = c(min(fgr.tiles.ncov3$ncov, -1.1), max(1.1, fgr.tiles.ncov3_2$ncov))
                xlab(paste0(chromosome, " (", tilesWidth, "nt tiles)"))+
                ylab("ncov (rpm)")

    # add scale bar
    pltCvrg <- pltCvrg +
        # Add the horizontal bar
        annotate("segment",
                x = (start(IRangesCoord) + 2),
                xend = (start(IRangesCoord) + 2) + scaleWidthKB * 1000,
                y = max(fgr.tiles.ncov3_2$ncov) * 0.9,
                yend = max(fgr.tiles.ncov3_2$ncov) * 0.9,
                color = "black",
                linewidth = 0.5) +
        # Add the text label
        annotate("text",
                x = (start(IRangesCoord) +2) + (scaleWidthKB*1000/2),
                y = max(fgr.tiles.ncov3_2$ncov) * 0.95,
                label = paste0(scaleWidthKB, " kb"),
                size = 2.5)

    # adapt to defined yRange
    if (length(yRange) == 0) {
        pltCvrg <- pltCvrg +  scale_y_continuous(breaks =
seq(round(min(fgr.tiles.ncov3$ncov, -1),roundBy),round(max(fgr.tiles.ncov3_2$ncov,
1),roundBy), by = yAxisBreaks))
    } else {
        pltCvrg <- pltCvrg +  scale_y_continuous(breaks = seq(round(yRange[1],roundBy),
round(yRange[2],roundBy), by = yRange[3])) + coord_cartesian(ylim = c(yRange[1],
yRange[2]))
    }
    return(list(pltCvrg = pltCvrg, fgr.tiles.ncov3 = fgr.tiles.ncov3, fgr.tiles.ncov3_2
= fgr.tiles.ncov3_2))

}


#GTF files for cluster can be filtered by its rank when in metacolumn a 'rank' column is
present.
#For transposable elements the family grouping is based on the column 'family_id'.
#For gene the height of the track is based on the column 'type' (CDS, UTR, exon, etc.)


#standarized function for coverage tracks, cluster and TE
allTracksPlotted <- function(piRNAs_from_Bam=c(),
                             chromosome, IRangesCoord,
                             gtfFiles=c(),
                             minRank = INF,
                             tilesWidth=100, xWidth=10000,
                             yRange=c(), #vector with min y-coordinate, max y-coordinate
and y-axis-breaks: c(y-min, y-max, breaks)
                             normalize = "rpm",
                             your_color_scale = NULL,
                             scaleWidthKB = ifelse(end(IRangesCoord) -
start(IRangesCoord) > 1000, 1, 0.2) # scale of the coverage track
                             ) {
```

```
    # set fgr
    fgr<-GRanges(seqnames = chromosome, ranges = IRangesCoord, strand = "-")
    fgr2<-GRanges(seqnames = chromosome, ranges = IRangesCoord, strand = "+")

    #perform coverage plot if bam file is given
    if (length(piRNAs_from_Bam) != 0) {
         cvrg <- coverageTracks(fgr, fgr2, piRNAs_from_Bam, chromosome, IRangesCoord,
tilesWidth, xWidth, yRange, normalize = normalize, scaleWidthKB = scaleWidthKB)
    } else {
        cvrg <- NULL
    }

    #retrieve cluster/TE/.. (from gtf) tracks
    if (length(gtfFiles) != 0) {
        #remove x-axis scale on coverage plot since it will be shown by piCs below (or
whichever gtf it is associated with)
        pltCvrg <- cvrg$pltCvrg + theme(axis.text.x = element_blank(),
                                        axis.ticks.x = element_blank(),
                                        axis.line.x = element_blank(),
                                        axis.title.x = element_blank()) # Remove x-axis
label completely

        ######## Tracks From GTF ########

        #define standard settings for gtf tracks
        clusterTracksStandard <- ggplot() + theme_classic() +
                                 coord_cartesian(xlim = c(start(IRangesCoord),
end(IRangesCoord))) +
                                 scale_x_continuous(breaks = seq(start(fgr), max(end(fgr)),
by = xWidth)) +
                                 theme(aspect.ratio = 0.05,
                                     axis.line.y = element_blank(),
                                     axis.text.y = element_blank(),
                                     axis.ticks.y = element_blank(),
                                     axis.title.y = element_text(angle = 0, vjust = 0.5),
                                     plot.title = element_text(hjust = 0.5),
                                     panel.grid.major.y = element_blank(),
                                     panel.grid.minor = element_blank(),
                                     axis.text.x = element_blank(),
                                     axis.ticks.x = element_blank(),
                                     axis.line.x = element_blank()) + # Remove x-axis label
completely
                                     xlab(NULL)

        #go through each given gtf file
        gtfNames <- names(gtfFiles)
        trackTotal <- list()
        gtfIt <- 0
        for (gtf in gtfFiles) {
            gtfIt <- gtfIt + 1

            #add TE-color code if applicable
            if ('class_id' %in% names(mcols(gtf))) {
                clusterTracksStandard <- clusterTracksStandard +
scale_colour_manual(values = your_color_scale)
            }
            has_gene_name <- 'gene_name' %in% names(mcols(gtf))
            has_gene_id <- 'gene_id' %in% names(mcols(gtf))
```

```
            #MINUS
            ovrlps_gtf1_minus <- findOverlaps(gtf, fgr)
            ovrlpsFeat_gtf1_minus <- as.data.frame(gtf[queryHits(ovrlps_gtf1_minus)])

                #Prep for clusters (include rank in metadata in gtf so
                if ('rank' %in% names(ovrlpsFeat_gtf1_minus)) {
                    ovrlpsFeat_gtf1_minus$rank <- as.integer(ovrlpsFeat_gtf1_minus$rank)
                    ovrlpsFeat_gtf1_minus <-
ovrlpsFeat_gtf1_minus[ovrlpsFeat_gtf1_minus$rank <= minRank, ]
                }

                #Prep for RMSK
                if ('class_id' %in% names(ovrlpsFeat_gtf1_minus)) {
                    TEtotalWidth <- sum(ovrlpsFeat_gtf1_minus$width)
                    sum_widths_by_familyMinus <- ovrlpsFeat_gtf1_minus %>%
                                                 group_by(class_id) %>%
                                                 summarise(total_width =
sum(width)/width(fgr))
                }

                cl_minus <- clusterTracksStandard + ylab(paste0("", gtfNames[gtfIt], "
(-)"))

                if ('class_id' %in% names(ovrlpsFeat_gtf1_minus)) {
                    #use segments for TE
                    cl_minus <- cl_minus +
                                    geom_segment(data=ovrlpsFeat_gtf1_minus, aes(x =
start, xend = end, y = 3.05, yend = 3.05, colour = class_id), linewidth = 5)

                } else {
                    # Draw rectangles (fallback to non-CDS/UTR height if 'type' is
missing)
                    if ('type' %in% names(ovrlpsFeat_gtf1_minus)) {
                        cl_minus <- cl_minus +
                            geom_rect(data=ovrlpsFeat_gtf1_minus[!grepl("CDS|UTR",
ovrlpsFeat_gtf1_minus$type),], aes(xmin = start, xmax = end, ymin = 4.9, ymax = 5.1),
fill="blue", col="black", linetype=1) +
                            geom_rect(data=ovrlpsFeat_gtf1_minus[grepl("exon",
ovrlpsFeat_gtf1_minus$type),], aes(xmin = start, xmax = end, ymin = 4, ymax = 6),
fill="blue", col="black", linetype=1) +
                            geom_rect(data=ovrlpsFeat_gtf1_minus[grepl("CDS",
ovrlpsFeat_gtf1_minus$type),], aes(xmin = start, xmax = end, ymin = 3, ymax = 7),
fill="blue", col="black", linetype=1) +
                            geom_rect(data=ovrlpsFeat_gtf1_minus[grepl("UTR",
ovrlpsFeat_gtf1_minus$type),], aes(xmin = start, xmax = end, ymin = 4, ymax = 6),
fill="blue", col="black", linetype=1)
                    } else {
                        cl_minus <- cl_minus +
                            geom_rect(data=ovrlpsFeat_gtf1_minus, aes(xmin = start, xmax
= end, ymin = 4.9, ymax = 5.1), fill="blue", col="black", linetype=1)
                    }

                    # Prepare and draw labels: prefer gene_name, fallback to gene_id; if
neither, skip
                    if (has_gene_name || has_gene_id) {
                        label_raw_minus <- if (has_gene_name)
ovrlpsFeat_gtf1_minus$gene_name else ovrlpsFeat_gtf1_minus$gene_id
                        label_clean_minus <- ifelse(grepl("rank_", label_raw_minus),
sub(".*rank_", "piC-", label_raw_minus), label_raw_minus)
                        labels_minus_df <- ovrlpsFeat_gtf1_minus %>%
```

```
                            mutate(.label = label_clean_minus) %>%
                            filter(!is.na(.label) & .label != "") %>%
                            group_by(.label) %>%
                            summarise(
                                start_min = if (all(is.na(start))) NA_real_ else
min(start, na.rm = TRUE),
                                end_max = if (all(is.na(end))) NA_real_ else max(end,
na.rm = TRUE),
                                .groups = "drop"
                            ) %>%
                            filter(!is.na(start_min) & !is.na(end_max)) %>%
                            mutate(x = (pmax(start_min, start(IRangesCoord)) +
pmin(end_max, end(IRangesCoord))) / 2)
                        cl_minus <- cl_minus +
                            geom_text(data=labels_minus_df, aes(x = x, y = 5, label =
.label), color="black", size=2.5)
                    }
                    cl_minus <- cl_minus + xlab(NULL)
                }

            #PLUS
            ovrlps_gtf1_plus <- findOverlaps(gtf, fgr2)
            ovrlpsFeat_gtf1_plus <- as.data.frame(gtf[queryHits(ovrlps_gtf1_plus)])
                #Prep for clusters (include rank in metadata in gtf so
                if ('rank' %in% names(ovrlpsFeat_gtf1_plus)) {
                    ovrlpsFeat_gtf1_plus$rank <- as.integer(ovrlpsFeat_gtf1_plus$rank)
                    ovrlpsFeat_gtf1_plus <-
ovrlpsFeat_gtf1_plus[ovrlpsFeat_gtf1_plus$rank <= minRank, ]
                }
                #Prep for RMSK
                if ('class_id' %in% names(ovrlpsFeat_gtf1_plus)) {
                    TEtotalWidth <- sum(ovrlpsFeat_gtf1_plus$width)
                    sum_widths_by_familyPlus <- ovrlpsFeat_gtf1_plus %>%
                                            group_by(class_id) %>%
                                            summarise(total_width =
sum(width)/width(fgr))
                }

                cl_plus <- clusterTracksStandard + ylab(paste0("", gtfNames[gtfIt], "
(+)"))

                   if ('family_id' %in% names(ovrlpsFeat_gtf1_plus)) {
                       #use segments for TE
                       cl_plus <- cl_plus +
                                        geom_segment(data=ovrlpsFeat_gtf1_plus, aes(x =
start, xend = end, y = 3.05, yend = 3.05, colour = class_id), linewidth = 5)

                    } else {
                        # Draw rectangles (fallback to non-CDS/UTR height if 'type' is
missing)
                        if ('type' %in% names(ovrlpsFeat_gtf1_plus)) {
                            cl_plus <- cl_plus +
                                geom_rect(data=ovrlpsFeat_gtf1_plus[!grepl("CDS|UTR",
ovrlpsFeat_gtf1_plus$type),], aes(xmin = start, xmax = end, ymin = 1.4, ymax = 1.6),
fill="red", col="black", linetype=1) +
                                geom_rect(data=ovrlpsFeat_gtf1_plus[grepl("exon",
ovrlpsFeat_gtf1_plus$type),], aes(xmin = start, xmax = end, ymin = 1, ymax = 2),
fill="red", col="black", linetype=1) +
                                geom_rect(data=ovrlpsFeat_gtf1_plus[grepl("CDS",
ovrlpsFeat_gtf1_plus$type),], aes(xmin = start, xmax = end, ymin = 0, ymax = 3),
```

```
fill="red", col="black", linetype=1) +
                                    geom_rect(data=ovrlpsFeat_gtf1_plus[grepl("UTR",
ovrlpsFeat_gtf1_plus$type),], aes(xmin = start, xmax = end, ymin = 1, ymax = 2),
fill="red", col="black", linetype=1)
                            } else {
                                cl_plus <- cl_plus +
                                    geom_rect(data=ovrlpsFeat_gtf1_plus, aes(xmin = start,
xmax = end, ymin = 1.4, ymax = 1.6), fill="red", col="black", linetype=1)
                            }

                            # Prepare and draw labels: prefer gene_name, fallback to
gene_id; if neither, skip
                            if (has_gene_name || has_gene_id) {
                                label_raw_plus <- if (has_gene_name)
ovrlpsFeat_gtf1_plus$gene_name else ovrlpsFeat_gtf1_plus$gene_id
                                label_clean_plus <- ifelse(grepl("rank_", label_raw_plus),
sub(".*rank_", "piC-", label_raw_plus), label_raw_plus)
                                labels_plus_df <- ovrlpsFeat_gtf1_plus %>%
                                    mutate(.label = label_clean_plus) %>%
                                    filter(!is.na(.label) & .label != "") %>%
                                    group_by(.label) %>%
                                    summarise(
                                        start_min = if (all(is.na(start))) NA_real_ else
min(start, na.rm = TRUE),
                                        end_max = if (all(is.na(end))) NA_real_ else
max(end, na.rm = TRUE),
                                        .groups = "drop"
                                    ) %>%
                                    filter(!is.na(start_min) & !is.na(end_max)) %>%
                                    mutate(x = (pmax(start_min, start(IRangesCoord)) +
pmin(end_max, end(IRangesCoord))) / 2)
                                cl_plus <- cl_plus +
                                    geom_text(data=labels_plus_df, aes(x = x, y = 1.5, label
= .label), color="black", size=2.5)
                            }
                            cl_plus <- cl_plus + xlab(NULL)
                        }
                #}


        #Include in last provided gtf file's minus track the x-scale with text, ticks
and line
        if (gtfIt == length(gtfFiles)) {
            cl_minus <- cl_minus + theme(axis.text.x = element_blank(),
                   axis.ticks.x = element_blank(),
                   axis.line.x = element_blank()) + xlab(paste0(chromosome, ":",
start(IRangesCoord), "-", end(IRangesCoord)))
        }

        #add to total list
        trackBoth <- list(trackMinus = cl_minus, trackPlus = cl_plus)
        trackTotal <- c(trackTotal, setNames(list(trackBoth), paste0("gtfNum", gtfIt)))
        }
    }
    else {
        #if no gtf file provided
        trackTotal <- NULL
        pltCvrg <- cvrg$pltCvrg
    }
    return(list(plotCoverageTrack = pltCvrg, trackAll = trackTotal, covg_plus =
```

```
cvrg$fgr.tiles.ncov3_2, covg_minus = cvrg$fgr.tiles.ncov3))
}
```

Mouse Pachytene piRNA Gene Targeting

Code for: Manuscript Figure 1 and Extended Data Figure 1

Introduction to files

Jupyter Notebook used R/4.4.2 and occasionally Bash.

Sample ID: 161922
GEO:
SRR:

5' Adapter: ACGACTTGGAATTCTCGGGTGCCAAGG
10 UMIs

Abbreviations
PCG: Protein Coding Genes
piC: piRNA Cluster
PG: Pseudogene

In this document, piCs are derived from MILI pachytene piCs published in Konstantinidou, et al. (2024, Cell Reports) and ranked by PICB column 'all_reads_primary_alignments_FPM'.
piCs were created with the Bioconductor package PICB.
For code purposes (e.g. for relating piRNAs to specific piCs) piRNAs that are not from piCs have rank 0.

Spin1-targeting
piC-as(Spin1) - MILI pachytene rank: 19, coordinates: chr9:67732316-67772950 (+), used in this document
piC-as(Spin1) - MIWI pachytene rank: 18, coordinates: chr9:67732666-67760175 (+)

Spin1-Gene coordinates in custom Protein Coding Gene Transcriptome: chr13:1698201-1702841 (+)
Spin1-Gene coordinates in mm10 genome (min and max of all annotated transcripts in RefSeq): chr13:51097934-51152562 (+)

Ago2-targeting
piC-as(Ago2) - MILI pachytene rank: 56, coordinates: chr4 123830771-123842320 (-), used in this document
piC-as(Ago2) - MIWI pachytene rank: 53, coordinates: chr4 123830736-123842320 (-)

Ago2-Gene coordinates in custom Protein Coding Gene Transcriptome: chr15:1357924-1365954 (-)
Ago2-Gene coordinates in mm10 genome (min and max of all annotated transcripts in RefSeq): chr15:73101625-73184947 (-)

Processing small RNA-seq

Mapping small RNA-seq data to mm10 genome

```
for entry in `ls raw_data/*ZL6_S7_R1.fastq.gz`; do     echo "$(basename "$entry")";
sbatch --mem=150g --cpus-per-task=32 --time=4:00:00
./mmu_pachytene_smallRNA_mapping_labVsUMIs2_wMiRNAout_wStarSeq.sh $(basename "$entry");
done
```

```
#!/bin/bash
-e

echo "Running script"

ml fastqc/0.11.9
ml cutadapt/4.4
ml STAR/2.7.10b
ml samtools/1.17

echo "Modules successfully loaded"

file_name=$1

echo "File to be pre-processed is ${file_name}"

file_ID=$(echo "$file_name" | sed 's/\.fastq\.gz$//;s/\.fastq$//')

echo "File ID of this file: ${file_ID}"

cutadapt -a ACGACTTGGAATTCTCGGGTGCCAAGG \
        --minimum-length 30 \
        -j 4 \
        -o prepro_data/"${file_ID}"_trimmed.fastq.gz \
        raw_data/"${file_name}" >
prepro_data/"${file_ID}"_trimmed_RemoveSmallRNA3pAdaptor_Report.txt

echo "Removed Illumina Small RNA 3' Adapter"

gunzip -c prepro_data/"${file_ID}"_trimmed.fastq.gz >
prepro_data/"${file_ID}"_trimmed.fastq
cat prepro_data/"${file_ID}"_trimmed.fastq | awk 'NR%4==2' | sort | uniq -c |  awk
'{OFS= "\n"; print ">"NR"-"$1,$2}' > prepro_data/"${file_ID}"_trimmed_collapsedA.fasta

cutadapt -u 8 -u -2 -o prepro_data/"${file_ID}"_trimmed_collapsed.fasta
prepro_data/"${file_ID}"_trimmed_collapsedA.fasta >
prepro_data/"${file_ID}"_RemoveUMI_Report.txt

echo "UMIs removed"

echo "Removing structural RNA, index exists"

STAR --runMode alignReads \
        --runThreadN 16 \
        --genomeDir ../mmu_background/background_wmiRNAs/fasta/backgroundDirWstar/ \
        --readFilesIn prepro_data/"${file_ID}"_trimmed_collapsed.fasta \
        --alignEndsType Local  \
    --outFilterMatchNmin 19 \
    --outFilterMultimapNmax 100 \
    --outFilterMismatchNmax 1 \
        --alignIntronMax 1 \
        --outReadsUnmapped Fastx \
```

```
    --outSAMattributes NH HI NM MD AS nM \
        --outFileNamePrefix prepro_data/structuralRemoval_miRoutwStar_"${file_ID}"_

echo "Mapped to structural RNA"

echo "Continue with those RNAs that did not map to structural RNAs"
cp prepro_data/structuralRemoval_miRoutwStar_"${file_ID}"_Unmapped.out.mate1 \

prepro_data/"${file_ID}"_trimmed_UMIextracted_last6ntRem_lengthFilter_structOut_miRoutwS
.fastq

fastqc -o prepro_data/FASTQC_prepro_data/
prepro_data/"${file_ID}"_trimmed_UMIextracted_last6ntRem_lengthFilter_structOut_miRoutwS
.fastq
echo "fastqc of finished prepro created"

cp
prepro_data/"${file_ID}"_trimmed_UMIextracted_last6ntRem_lengthFilter_structOut_miRoutwS
.fastq \

cleaned_data/"${file_ID}"_trimmed_UMIextracted_last6ntRem_lengthFilter_structOut_miRoutw
S.fastq

echo "finished prepro - moved to cleaned_data"

echo "Mouse genome is already indexed in ../mmu_referenceGenome/GRCm38.p6.genDir/"
echo "Start ${file_ID} mapping"

STAR --runMode alignReads \
        --runThreadN 23 \
        --genomeDir ../mmu_referenceGenome/GRCm38.p6.genDir/ \
        --readFilesIn
cleaned_data/"${file_ID}"_trimmed_UMIextracted_last6ntRem_lengthFilter_structOut_miRoutw
S.fastq  \
        --alignEndsType EndToEnd \
        --outSAMattriD  butes All \
        --outSAMtype BAM SortedByCoordinate \
        --limitBAMsortRAM 40000000000 \
        --alignIntronMax 1 \
        --alignSoftClipAtReferenceEnds No \
        --outFilterMismatchNmax 1 \
        --winAnchorMultimapNmax 100 \
        --outFilterMultimapNmax 100 \
        --outReadsUnmapped Fastx \
    --outFileNamePrefix
STAROutput/smallRNAs_pach_"${file_ID}"/"${file_ID}"_trimmed_UMIcollapsed_structOut_miRou
twS_

echo "Mapped to mouse genome"

samtools index
STAROutput/smallRNAs_pach_"${file_ID}"/"${file_ID}"_trimmed_UMIcollapsed_structOut_miRou
twS_Aligned.sortedByCoord.out.bam
echo "Indexed STARoutput bam file"

echo "Done."
```

Preparation: PCG-Transcriptome

Protein-coding gene transcriptome creation with ../scripts/buildTranscriptomes.R

Mapping to PCG-transcriptome

Create fasta with all mapped reads including rank and coordinates of mapping

```
suppressPackageStartupMessages({
    library("GenomicRanges")
    library("Biostrings")
    library("BSgenome.Mmusculus.UCSC.mm10")
    library(PICB)
})
```

```
# Use PICBload to load piRNAs
bam_file <-
"Mouse_161922/Mouse_161922_testes_small_RNAs_ZL6_S7_R1_trimmed_UMIcollapsed_structOut_miRoutwS_Aligned.sortedByCoord.out.bam"
myGenome <- "BSgenome.Mmusculus.UCSC.mm10"

alignWT_161922 <- PICBload(
    BAMFILE = bam_file,
    REFERENCE.GENOME = myGenome,
    GET.ORIGINAL.SEQUENCE = TRUE,
    VERBOSE = 0,

)

saveRDS(alignWT_161922, file =
"Mouse_161922/Mouse_161922_testes_small_RNAs_ZL6_S7_R1_trimmed_UMIcollapsed_structOut_miRoutwS_Aligned.PICBloadWseqs.RDS")
```

```
alignWT_161922 <-
readRDS("Mouse_161922/Mouse_161922_testes_small_RNAs_ZL6_S7_R1_trimmed_UMIcollapsed_structOut_miRoutwS_Aligned.PICBloadWseqs.RDS")
alignWT_161922_unique_multiPrim <- c(alignWT_161922$unique,
alignWT_161922$multi.primary)
length(alignWT_161922_unique_multiPrim)
```

```
70317338
```

```
#get cluster coordinates (ranked!) from prev. publication (mouse MILI pachytene, Konstantinidou et al. (2024) in Cell Reports)
load("../../../OneDrive/General/mmu_piRNA_clusters_byThenia/MILIclusters_pachytene.RData") #MILI_prepach_regions_overl_genes
MILIclusters_pach <- MILIclusters$clusters

MILIclusters_pach <- MILIclusters_pach[order(-
MILIclusters_pach$all_reads_primary_alignments_FPM), ]
MILIclusters_pach$rankByAllReadsPrimaryAlignmentsFPM <- seq_along(MILIclusters_pach)
length(MILIclusters_pach)
```

```
4188
```

```
# Find overlaps between piRNAs and piRNA clusters
piRNAs_fromPiC_f0 <- findOverlaps(alignWT_161922_unique_multiPrim, MILIclusters_pach,
ignore.strand=FALSE)

mcols(alignWT_161922_unique_multiPrim)$corr_piC_rankByAllReadsPrimaryAlignmentsFPM <- 0

mcols(alignWT_161922_unique_multiPrim)$corr_piC_rankByAllReadsPrimaryAlignmentsFPM[query
Hits(piRNAs_fromPiC_f0)] <-
mcols(MILIclusters_pach[subjectHits(piRNAs_fromPiC_f0)])$rankByAllReadsPrimaryAlignments
FPM
length(alignWT_161922_unique_multiPrim)
```

```
70317338
```

```
#add to readname rank_chr_startPos_strand
seq <- alignWT_161922_unique_multiPrim$seq
names(seq) <- paste0(names(alignWT_161922_unique_multiPrim), "_rank",
alignWT_161922_unique_multiPrim$corr_piC_rankByAllReadsPrimaryAlignmentsFPM, "_",
seqnames(alignWT_161922_unique_multiPrim), "_", start(alignWT_161922_unique_multiPrim),
strand(alignWT_161922_unique_multiPrim), "_NH", alignWT_161922_unique_multiPrim$NH)

# save sequences of piRNAs with readname infos in fasta format
writeXStringSet(seq,
file="Mouse_161922/Mouse_161922_testes_small_RNAs_ZL6_S7_R1_trimmed_UMIcollapsed_structO
ut_miRoutwS_Aligned.PICBloadWseqs.primAlignWinfo.fasta")
```

Mapping to PCG-Exon-Transcriptome

Indexing Transcriptome

Bash

```
#BASH, generateGenome
STAR --runMode genomeGenerate \
       --genomeDir
../../../OneDrive/General/mmu_referenceGenome/PCG_transcriptome/transcriptomeDir \
       --genomeSAindexNbases 6 \
       --genomeFastaFiles
../../../OneDrive/General/mmu_referenceGenome/PCG_transcriptome/mm10_PCGtranscriptome_co
llapsed_prioritizedCDS3UTR5UTR_allgenes.fasta \
       --limitGenomeGenerateRAM 34173092106\
       --runThreadN 23
```

```
Mar 21 14:55:36 ..... started STAR run
Mar 21 14:55:36 ... starting to generate Genome files
Mar 21 14:55:37 ... starting to sort Suffix Array. This may take a long time...
Mar 21 14:55:37 ... sorting Suffix Array chunks and saving them to disk...
Mar 21 14:56:24 ... loading chunks from disk, packing SA...
Mar 21 14:56:25 ... finished generating suffix array
Mar 21 14:56:25 ... generating Suffix Array index
Mar 21 14:56:25 ... completed Suffix Array index
Mar 21 14:56:25 ... writing Genome to disk ...
Mar 21 14:56:25 ... writing Suffix Array to disk ...
Mar 21 14:56:25 ... writing SAindex to disk
Mar 21 14:56:25 ..... finished successfully
```

Mapping to Transcriptome

Bash

```
addFastaChange=""
addMappingChange="clip5pNbases1_Extend5pOfRead1_minMatch19"
```

```
#bash
input_fasta="Mouse_161922/Mouse_161922_testes_small_RNAs_ZL6_S7_R1_trimmed_UMIcollapsed_
structOut_miRoutwS_Aligned.PICBloadWseqs.primAlignWinfo.fasta"

#minimum Match of 19 nt, to restrict soft-clipping
STAR --runMode alignReads \
    --runThreadN 20 \
    --genomeDir
../../../OneDrive/General/mmu_referenceGenome/PCG_transcriptome/transcriptomeDir/ \
    --readFilesIn $input_fasta \
    --clip5pNbases 1 \
    --alignEndsType Extend5pOfRead1 \
    --outSAMattributes All \
    --outSAMtype BAM SortedByCoordinate \
    --limitBAMsortRAM 20000000000 \
    --alignIntronMax 1 \
    --alignSoftClipAtReferenceEnds No \
    --outFilterMismatchNmax 1 \
    --outFilterMatchNmin 19 \
    --winAnchorMultimapNmax 100 \
    --outFilterMultimapNmax 100 \
    --outReadsUnmapped Fastx \
    --outFileNamePrefix
pachTargetingGenes_PCG/Mouse_161922/PCG_Mouse_161922_testes_small_RNAs_ZL6_S7_R1_trimmed
_UMIcollapsed_structOut_miRoutwS_Aligned.PICBloadWseqs.primAlignWinfo.${addFastaChange}.
PCGtranscriptome_${addMappingChange}_

echo "Mapped to PCG_EXON transcriptome"
```

```
        Mar 24 10:03:10 ..... started STAR run
        Mar 24 10:03:10 ..... loading genome
        Mar 24 10:03:10 ..... started mapping
        Mar 24 10:05:21 ..... started sorting BAM
        Mar 24 10:05:24 ..... finished successfully
        Mapped to PCG_EXON transcriptome
```

```
suppressPackageStartupMessages({library(Rsamtools)})
#indexBam for every Bam in the folder pachTargetingGenes_PCG/Mouse_161922/ in R
for (bam in list.files("pachTargetingGenes_PCG/Mouse_161922", pattern="\\.bam$",
full.names=TRUE)) {
    indexBam(bam)
}
```

Which genes are targeted by pachytene piRNAs?

Prep - Load alignments and gene annotations

```
suppressPackageStartupMessages({
    library(GenomicRanges)
    library(Rsamtools)
```

```
    library(GenomicAlignments)
    library(tidyr)
    library(dplyr)
    library(ggplot2)
    library(ggrepel)
    library(patchwork)
    library(Biostrings)
    library(UniProt.ws)
    library(scales)
    library(SVbyEye)
    library(Rsubread)
    library(DESeq2)
    library(PICB)
    library(openxlsx)
    library(BSgenome.Mmusculus.UCSC.mm10)
})
source("../scripts/plotCoverage.R")
```

```
# Load BAM file of piRNAs targeting protein coding genes
bamPCG_dir <-
"../data/bam/mmuToTranscriptome/PCG_Mouse_161922_testes_small_RNAs_ZL6_S7_R1_trimmed_UMI
collapsed_structOut_miRoutwS_Aligned.PICBloadWseqs.primAlignWinfo..PCGtranscriptome_clip
5pNbases1_Extend5pOfRead1_minMatch19_Aligned.sortedByCoord.out.bam"

bam <- BamFile(bamPCG_dir)
# exclude both secondary alignments and supplementary alignments
fields <- scanBamWhat()
primary_flag <- scanBamFlag(isSecondary = FALSE, isSupplementary = FALSE)
param <- ScanBamParam(flag = primary_flag,
    what=c('qname','flag','rname','strand','pos','qwidth', 'cigar', 'seq'),
    tag=c('NH', "MD"))

ga_all_alignments <- readGAlignments(bam, param = param)
PCG_total_reads <- length(ga_all_alignments)
ga_alignments <- ga_all_alignments[mcols(ga_all_alignments)$NH == 1] #filter by unique
alignments
PCG_unique_reads <- length(ga_alignments)

gr_alignments <- makeGRangesFromDataFrame(as.data.frame(ga_alignments),
keep.extra.columns=TRUE)

mcols(gr_alignments) <- mcols(gr_alignments)[ , !(colnames(mcols(gr_alignments)) %in%
c("njunc", "strandInfo", "rname", "strand.1", "pos", "qwidth.1", "cigar.1", "qual"))]
```

```
# load annotation of the protein coding genes
geneAnnotation_PCG_EXON_dir <-
"../data/annotations/customTranscriptomes/mm10_PCGtranscriptome_collapsed_prioritizedCDS
3UTR5UTR_allgenes.gtf"

#geneAnnotation_PCG_EXON <- rtracklayer::import(geneAnnotation_PCG_EXON_dir)

# Import GTF file without rtracklayer (issues with installation)
read_gtf <- function(file_path) {
  # Read the GTF file - changed start/end to numeric first, then convert to integer
  gtf_data <- read.table(file_path, sep="\t", quote="",
                         col.names=c("seqname", "source", "feature", "start", "end",
                                     "score", "strand", "frame", "attribute"),
                         colClasses=c("character", "character", "character", "numeric",
```

```
"numeric",
                                    "character", "character", "character", "character"))

  # Convert coordinates to integer after reading
  gtf_data$start <- as.integer(gtf_data$start)
  gtf_data$end <- as.integer(gtf_data$end)

  # Function to extract attributes
  extract_attribute <- function(attr, key) {
    val <- sub(paste0(".*", key, "\\s+\"?([^;\"]+)\"?.*"), "\\1", attr)
    ifelse(val == attr, NA, val)
  }

  # Extract common attributes
  gtf_data$gene_id <- extract_attribute(gtf_data$attribute, "gene_id")
  gtf_data$transcript_id <- extract_attribute(gtf_data$attribute, "transcript_id")
  gtf_data$gene_name <- extract_attribute(gtf_data$attribute, "gene_name")

  return(gtf_data)
}


geneAnnotation_PCG <- read_gtf(geneAnnotation_PCG_EXON_dir)

geneAnnotation_PCG <- makeGRangesFromDataFrame(geneAnnotation_PCG,
                                               keep.extra.columns = TRUE,
                                               seqnames.field = "seqname",
                                               start.field = "start",
                                               end.field = "end",
                                               strand.field = "strand")

geneAnnotation_PCG_gene <- geneAnnotation_PCG[geneAnnotation_PCG$feature == "gene"]
```

```
# load gene coordinates of collapsed, prioritized (CDS > 3UTR > 5UTR) genes in mm10
genome
genes_dir <- "../data/annotations/mm10_collapsed_prioritizedCDS3UTR5UTR.gtf"
genes <- rtracklayer::import(genes_dir)
```

```
#piRNA cluster coordinates from Konstantinidou et al. (2024) in Cell Reports and rank
load("../data/annotations/MILIclusters_pachytene.RData")
MILIclusters_pach <- MILIclusters$clusters

MILIclusters_pach <- MILIclusters_pach[order(-
MILIclusters_pach$all_reads_primary_alignments_FPM), ]
MILIclusters_pach$rankByAllReadsPrimaryAlignmentsFPM <- seq_along(MILIclusters_pach)
MILIclusters_pach <- keepStandardChromosomes(MILIclusters_pach)
paste0("Number of piRNA clusters in MILI pachytene by PICB: ",
length(MILIclusters_pach))
```

'Number of piRNA clusters in MILI pachytene by PICB: 4188'

```
# load alignments to mm10 genome
gr_mm10 <- PICBload(
    BAMFILE =
"../data/bam/Mouse_161922_testes_small_RNAs_ZL6_S7_R1_trimmed_UMIcollapsed_structOut_miR
outwS_Aligned.sortedByCoord.out.bam",
```

```
    REFERENCE.GENOME = "BSgenome.Mmusculus.UCSC.mm10",
    GET.ORIGINAL.SEQUENCE = TRUE, VERBOSE = FALSE, WHAT = c("flag", "cigar"))

gr_mm10_prim <- c(gr_mm10$unique, gr_mm10$multi.primary)
genome_total_reads <- length(gr_mm10_prim)
message("Number of reads mapped to mm10 (primary alignments): ", genome_total_reads)
gr_mm10_prim <- keepStandardChromosomes(gr_mm10_prim)

gr_mm10 <- gr_mm10$unique
genome_unique_reads <- length(gr_mm10)
message("Number of reads mapped to mm10 (unique alignments only): ",
genome_unique_reads)
gr_mm10 <- keepStandardChromosomes(gr_mm10)
gr_mm10$qname <- names(gr_mm10)
```

```
Number of reads mapped to mm10 (primary alignments): 70317338

Number of reads mapped to mm10 (unique alignments only): 64104551
```

Genes targeted by piRNAs (Fig 1a)

```
# Initialize result dataframe with all genes
topPiCtarget_df <- data.frame(
    geneName = geneAnnotation_PCG_gene$gene_name,
    topContribPiCrank = "0",
    topPercentage = 0,
    bypiCtargeting_percentage = 0,
    top_piC_percentage = 0,
    cisTargeting = FALSE,
    stringsAsFactors = FALSE
)

# Create function to get top contributing piC and percentage
get_top_piC_info <- function(overlaps) {
    if (length(overlaps) == 0) {
        return(c("0", 0, 0, 0))
    }
    # Extract rank information using vectorized operations
    ranks <- sub("_|_", "", sub("rank|_", "", regmatches(overlaps$qname,
regexpr("rank(.*?)_", overlaps$qname))))
    rank_table <- table(ranks)
    top_rank <- names(which.max(rank_table))
    top_percentage <- max(rank_table) / length(overlaps)
    # Get the percentage of targeting by piC
    bypiCtargeting_percentage <- sum(rank_table[names(rank_table) != "0"]) /
length(overlaps)

    if (top_rank == "0") {
        sorted_counts <- sort(rank_table, decreasing = TRUE)
        second_largest <- sorted_counts[2]
        top_piC_percentage <- second_largest / length(overlaps)
    } else {
        top_piC_percentage <- top_percentage
    }
    if (is.na(top_piC_percentage)) {
        top_piC_percentage <- 0
    }

    return(c(top_rank, top_percentage, top_piC_percentage, bypiCtargeting_percentage))
```

```
}

# Get all overlaps at once
all_overlaps <- findOverlaps(invertStrand(geneAnnotation_PCG_gene), gr_alignments)

# Split overlaps by gene
overlaps_by_gene <- split(gr_alignments[subjectHits(all_overlaps)],
                          queryHits(all_overlaps))

# Apply function to each gene's overlaps
results <- lapply(overlaps_by_gene, get_top_piC_info)

# Update only the rows that have overlaps
genes_with_overlaps <- as.numeric(names(overlaps_by_gene))
topPiCtarget_df$totalPiRNAcount <- countOverlaps(invertStrand(geneAnnotation_PCG_gene),
gr_alignments)
topPiCtarget_df$topContribPiCrank[genes_with_overlaps] <- sapply(results, `[`, 1)
topPiCtarget_df$topPercentage[genes_with_overlaps] <- as.numeric(sapply(results, `[`,
2))
topPiCtarget_df$bypiCtargeting_percentage[genes_with_overlaps] <-
as.numeric(sapply(results, `[`, 3))
topPiCtarget_df$top_piC_percentage[genes_with_overlaps] <- as.numeric(sapply(results,
`[`, 4))
rownames(topPiCtarget_df) <- topPiCtarget_df$geneName

# After updating topContribPiCrank and topPercentage, update cisTargeting
inverted_clusters <- invertStrand(MILIclusters_pach)
topContribPiCrank_numeric <- as.numeric(as.character(topPiCtarget_df$topContribPiCrank))

for (i in seq_len(nrow(topPiCtarget_df))) {
    geneNameI <- topPiCtarget_df$geneName[i]
    gene_overlaps <- subsetByOverlaps(
        inverted_clusters,
        genes[genes$gene_id == geneNameI])$rankByAllReadsPrimaryAlignmentsFPM
    topPiCtarget_df$cisTargeting[i] <- topContribPiCrank_numeric[i] %in% gene_overlaps
}
```

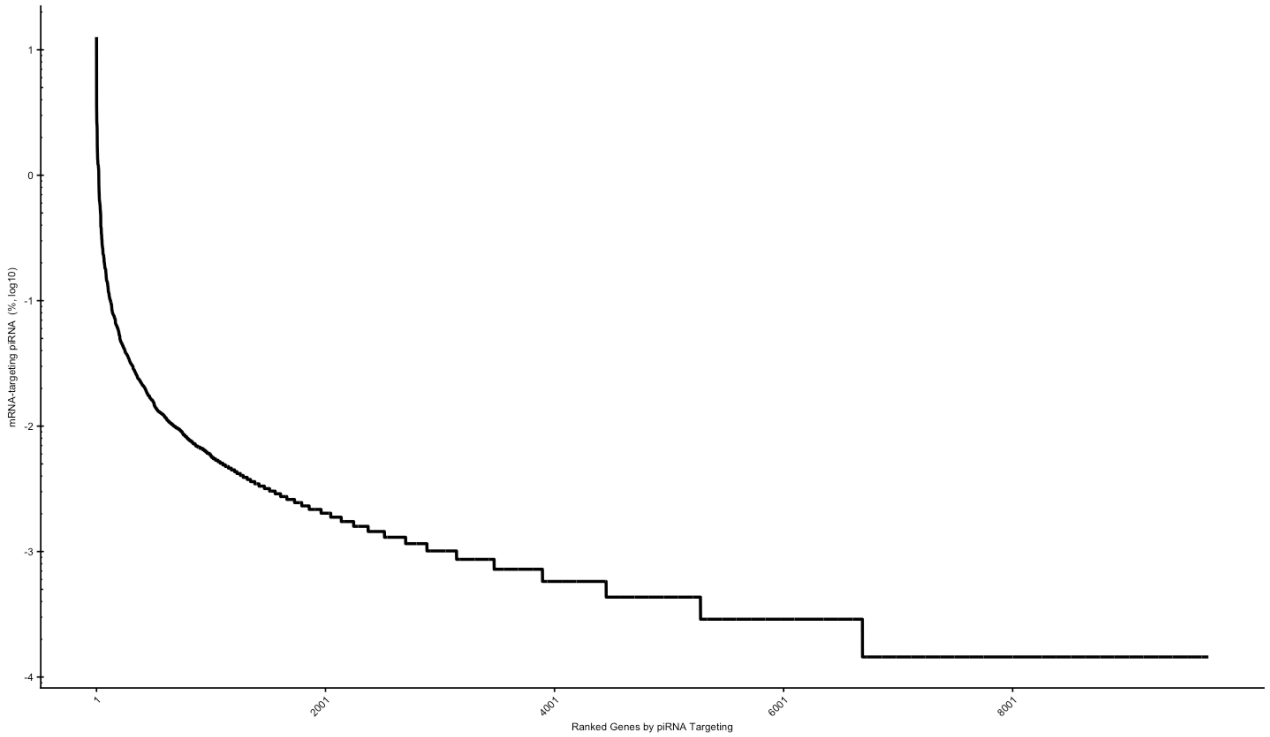
```
# Filter for non-cis targeted genes and order by totalPiRNAcount
targetingByPiRNAsSbO_sorted_total_tra <- topPiCtarget_df[topPiCtarget_df$cisTargeting ==
FALSE,]
message("Number of genes in antisense orientation to piRNA clusters and therefore
removed: ", nrow(topPiCtarget_df) - nrow(targetingByPiRNAsSbO_sorted_total_tra))
targetingByPiRNAsSbO_sorted_total_tra <-
targetingByPiRNAsSbO_sorted_total_tra[order(targetingByPiRNAsSbO_sorted_total_tra$totalP
iRNAcount, decreasing=TRUE),]

# Calculate percentage of targeting piRNAs for each gene
readsTargetingPCGs <- sum(targetingByPiRNAsSbO_sorted_total_tra$totalPiRNAcount)
targetingByPiRNAsSbO_sorted_total_tra$totalPiRNAperc <-
(targetingByPiRNAsSbO_sorted_total_tra$totalPiRNAcount/readsTargetingPCGs)*100
targetingByPiRNAsSbO_sorted_total_tra$targetedRank <-
1:nrow(targetingByPiRNAsSbO_sorted_total_tra)
```

```
Number of genes in antisense orientation to piRNA clusters and therefore remove
d: 157
```

```r
# establish targeting threshold
topPercentageCS <- 0.7
cumsum_totalPiRNAcount <- cumsum(targetingByPiRNAsSbO_sorted_total_tra$totalPiRNAcount)
cumsum_targetingThreshold <- readsTargetingPCGs*topPercentageCS
print(paste0(topPercentageCS*100, "% threshold value: ", cumsum_targetingThreshold))
genes_targetingThreshold <- which(cumsum_totalPiRNAcount >= cumsum_targetingThreshold)
[1]
paste0("", genes_targetingThreshold)
```

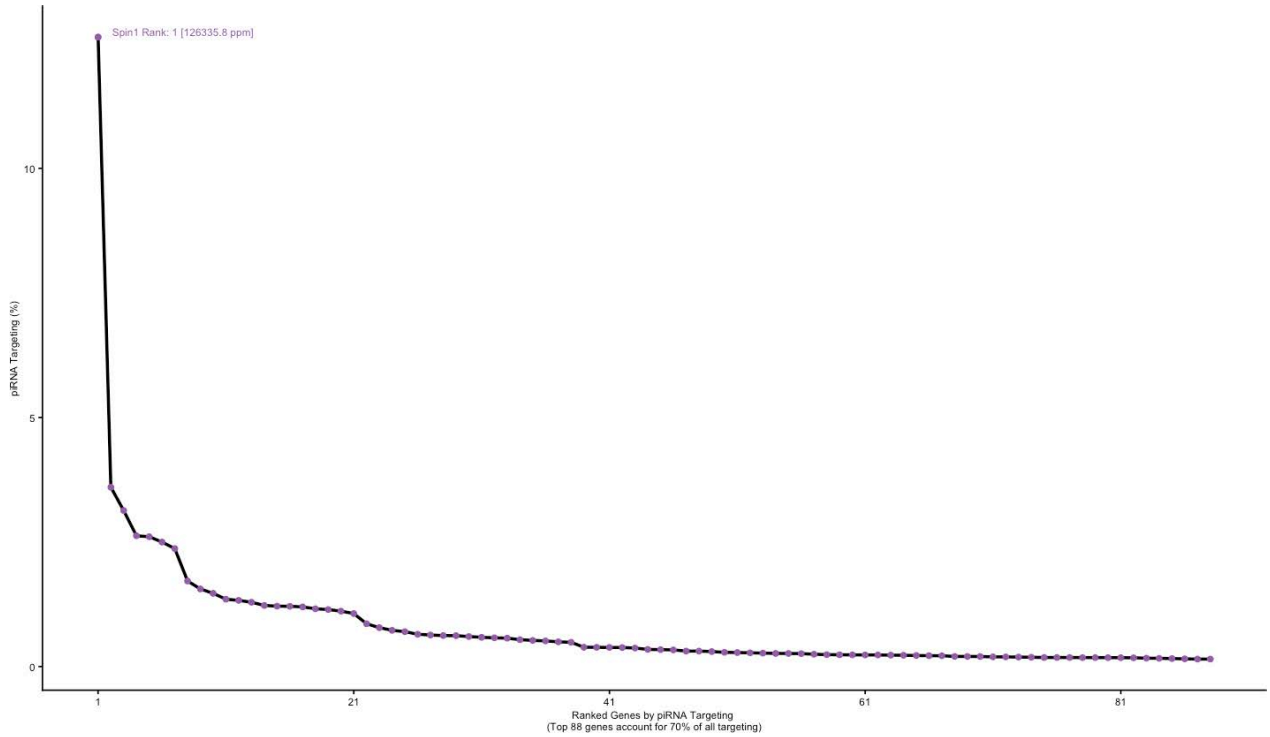
```
[1] "70% threshold value: 484348.2"
'88'
```

```r
options(repr.plot.width=12, repr.plot.height=7)
plot_1a_full <-
ggplot(targetingByPiRNAsSbO_sorted_total_tra[targetingByPiRNAsSbO_sorted_total_tra$total
PiRNAcount != 0,],
    aes(x = targetedRank, y = log10(totalPiRNAperc), group = 1)) +
    geom_vline(xintercept = 0:genes_targetingThreshold, color = "#ffffff", alpha = 0.3)
+
    scale_x_continuous(breaks = seq(1,
nrow(targetingByPiRNAsSbO_sorted_total_tra[targetingByPiRNAsSbO_sorted_total_tra$totalPi
RNAcount != 0,]), by = 2000)) +
    geom_line(linewidth = 1) +
    theme_classic() +
    annotation_logticks(base = 10, sides = "l", short = unit(0.02, "cm"), mid =
unit(0.04, "cm"), long = unit(0.06, "cm")) +
    theme(
        axis.text.x = element_text(angle = 45, hjust = 1, size = 7),
        axis.text.y = element_text(size = 7),
        axis.title.x = element_text(size = 7),
        axis.title.y = element_text(size = 7)
    ) +
    labs(
        x = "Ranked Genes by piRNA Targeting",
        y = "mRNA-targeting piRNA  (%, log10)"
    )

plot_1a_full
```




```
options(repr.plot.width=12, repr.plot.height=7)

# Get the data for specified genes
highlight_genes <- c('Spin1')
gene_data <- targetingByPiRNAsSbO_sorted_total_tra[
    targetingByPiRNAsSbO_sorted_total_tra$geneName %in% highlight_genes,]

# Create plot with highlighted genes and corrected ranks
plot_1aInset <-
ggplot(targetingByPiRNAsSbO_sorted_total_tra[1:genes_targetingThreshold,],
    aes(x = targetedRank, y = totalPiRNAperc, group = 1)) +
    geom_line(linewidth = 1, color = "black") +
    geom_point(color = "#9662A9", size = 1.5) +
    scale_x_continuous(breaks = seq(1, nrow(targetingByPiRNAsSbO_sorted_total_tra), by =
20)) +
    # Add highlighted points with different color/size to make them stand out
    geom_point(data = gene_data, color = "#9662A9", size = 1.5) +
    # Add labels for highlighted genes
    geom_text(data = gene_data,
              aes(label = sapply(geneName, function(g) {
                  count <-
targetingByPiRNAsSbO_sorted_total_tra$totalPiRNAcount[targetingByPiRNAsSbO_sorted_total_
tra$geneName == g]
                  ppm <- round((count / readsTargetingPCGs) * 1e6, 1)
                  paste0(g, " Rank: ",
                         match(g, targetingByPiRNAsSbO_sorted_total_tra$geneName),
                         " [", ppm, " ppm]")
              })),
              color = "#9662A9",
              vjust = -0.2,  # Single value for single gene
              hjust = -0.1,  # Single value for single gene
              size = 2.5) +
    theme_classic() +
    theme(
        axis.text.x = element_text(size = 7),
```

```
        axis.text.y = element_text(size = 7),
        axis.title.x = element_text(size = 7),
        axis.title.y = element_text(size = 7)
    ) +
    labs(
        x = paste0("Ranked Genes by piRNA Targeting \n(Top ", genes_targetingThreshold,
" genes account for ", topPercentageCS*100, "% of all targeting)"),
        y = "piRNA Targeting (%)"
    )

plot_1aInset
```



Scatter plot of mRNA target length and the log10-transformed percentage of targeting piRNAs (Fig 1c)

Get Pseudogene Information and associate with piRNA clusters and genes they target

```
#load pseudogenes, download by UCSC
pseudogenes_dir <- "../data/annotations/mm10_retroGenesV6.gtf"
pseudogenes <- rtracklayer::import(pseudogenes_dir)
```

```
# prefilter piRNA clusters that are generally targeting PCG genes
maxPiC <-
max(as.integer(targetingByPiRNAsSbO_sorted_total_tra[1:genes_targetingThreshold,]$topCon
tribPiCrank))
message("Only considering top ", maxPiC, " piRNA clusters since top ",
genes_targetingThreshold, " targeted genes only are targeted by that max rank.")
MILIclusters_pachSubset <-
MILIclusters_pach[MILIclusters_pach$rankByAllReadsPrimaryAlignmentsFPM <= maxPiC]
pseudogenesRGOvrlp <- subsetByOverlaps(pseudogenes,
invertStrand(MILIclusters_pachSubset))
pseudogenesRGOvrlp$parent_transcript <- sub("\\..*", "", pseudogenesRGOvrlp$gene_id)
unique_transcripts <- unique(pseudogenesRGOvrlp$parent_transcript)
```

```
Only considering top 127 piRNA clusters since top 88 targeted genes only are tar
geted by that max rank.
```

```r
# Extract gene names from parent_transcripts of pseudogenes (so they can be matched with
our gene annotations)
# load UniProt.ws for mouse
mmuUp <- UniProt.ws(10090)
mmuUp

# Get gene_name for a parent_transcript
columns = c('gene_primary')
getGeneName <- function(transcript) {
    if (grepl("^NM", transcript)){
        keytype = c("RefSeq_Nucleotide")
    } else {
        keytype = c("EMBL-GenBank-DDBJ")
    }

    result <- select(mmuUp, keys = transcript, keytype = keytype, columns = columns)

    if (nrow(result) > 0) {
        return(result$Gene.Names..primary.[1])
    } else {
        return(NA)
    }
}

# Create a mapping of parent_transcript to gene_name
transcript_to_gene <- sapply(unique_transcripts, getGeneName)

# Update parent_id in pseudogenes
pseudogenesRGOvrlp$parent_id <- transcript_to_gene[pseudogenesRGOvrlp$parent_transcript]
```

```
Error in curl::curl_fetch_memory(url, handle = handle): Could not resolve hostna
me [rest.uniprot.org]:
Could not resolve host: rest.uniprot.org
Traceback:

1. queryUniProt(query = paste0("taxonomy_id:", taxId), fields = c("accession",
 .     "organism_name"), n = 25, pageSize = 25)
2. .uniprotPages(FUN = .searchPaged, query = query, fields = fields,
 .     collapse = collapse, n = n, pageSize = pageSize)
3. FUN(url = url, ..., pageSize = pageSize)
4. httpcache::GET(url = url, query = list(query = paste(query, collapse = collap
se),
 .     fields = paste(fields, collapse = ","), format = "tsv", size = pageSize))
5. httr::GET(url, ...)
6. request_perform(req, hu$handle$handle)
7. request_fetch(req$output, req$url, handle)
8. request_fetch.write_memory(req$output, req$url, handle)
9. curl::curl_fetch_memory(url, handle = handle)
10. raise_libcurl_error(6L, "Could not resolve hostname", "Could not resolve hos
t: rest.uniprot.org",
  .     "https://rest.uniprot.org/uniprotkb/search?query=taxonomy_id%3A10090&fie
lds=accession%2Corganism_name&format=tsv&size=25",
  .     NULL)
11. stop(e)
```

```
#look up NAs in parent_id by hand and replace with actual gene name (of parent from PG)
unique(pseudogenesRGOvrlp[is.na(pseudogenesRGOvrlp$parent_id)]$parent_transcript)

#replace parent_id of parent_transcripts manually
pseudogenesRGOvrlp[pseudogenesRGOvrlp$parent_transcript == "AK209531"]$parent_id <-
"Arpc5"
pseudogenesRGOvrlp[pseudogenesRGOvrlp$parent_transcript == "NR_027488"]$parent_id <-
"Senp2"
pseudogenesRGOvrlp[pseudogenesRGOvrlp$parent_transcript == "BC006068"]$parent_id <-
"Rpl10"
pseudogenesRGOvrlp[pseudogenesRGOvrlp$parent_transcript == "AK076357"]$parent_id <-
"Ddx11"
```

'AK209531' · 'NR_027488' · 'BC006068' · 'AK076357'

```
# Find overlaps
overlaps <- findOverlaps(pseudogenesRGOvrlp, invertStrand(MILIclusters_pachSubset))

# Extract ranks based on overlaps
ranks <- rep(NA, length(pseudogenesRGOvrlp))
ranks[queryHits(overlaps)] <-
MILIclusters_pachSubset$rankByAllReadsPrimaryAlignmentsFPM[subjectHits(overlaps)]

# Add rank to pseudogenesRGOvrlp
pseudogenesRGOvrlp$rankByAllReadsPrimaryAlignmentsFPM <- ranks
```

```
# Get unique pairs
unique_pairs <- unique(data.frame(
    parent_id = mcols(pseudogenesRGOvrlp)$parent_id,
    rank = mcols(pseudogenesRGOvrlp)$rankByAllReadsPrimaryAlignmentsFPM
))

# Sort by cluster_rank if desired
unique_pairs <- unique_pairs[order(unique_pairs$rank), ]

unique_pairs$geneTop250targeted <- unique_pairs$parent_id %in% topPiCtarget_df$geneName

# Add Unc119b row (unannotated Pseudogene)
unique_pairs <- rbind(
    unique_pairs,
    data.frame(
        parent_id = "Unc119b",
        rank =
targetingByPiRNAsSbO_sorted_total_tra[targetingByPiRNAsSbO_sorted_total_tra$geneName ==
"Unc119b",]$topContribPiCrank,
        geneTop250targeted = TRUE,
        stringsAsFactors = FALSE
    )
)
```

A data.frame: 17 x 3

	parent_id	rank	geneTop250targeted
	<chr>	<chr>	<lgl>
207	Ipmk	6	TRUE

	parent_id	rank	geneTop250targeted
	<chr>	<chr>	<lgl>
354	Spin1	19	TRUE
385	Crxos	23	TRUE
421	Phpln1	25	TRUE
361	Rpl10	28	TRUE
419	Slc25a4	41	TRUE
43	Arhgap20	42	TRUE
305	Senp2	43	TRUE
1	Ago2	56	TRUE
9	Arpc5	56	TRUE
435	Mark2	65	TRUE
156	Stambp	67	TRUE
374	Ddx11	70	TRUE
35	Arpc5	110	TRUE
403	Rpl7a	113	TRUE
328	Senp2	120	TRUE
11	Unc119b	46	TRUE

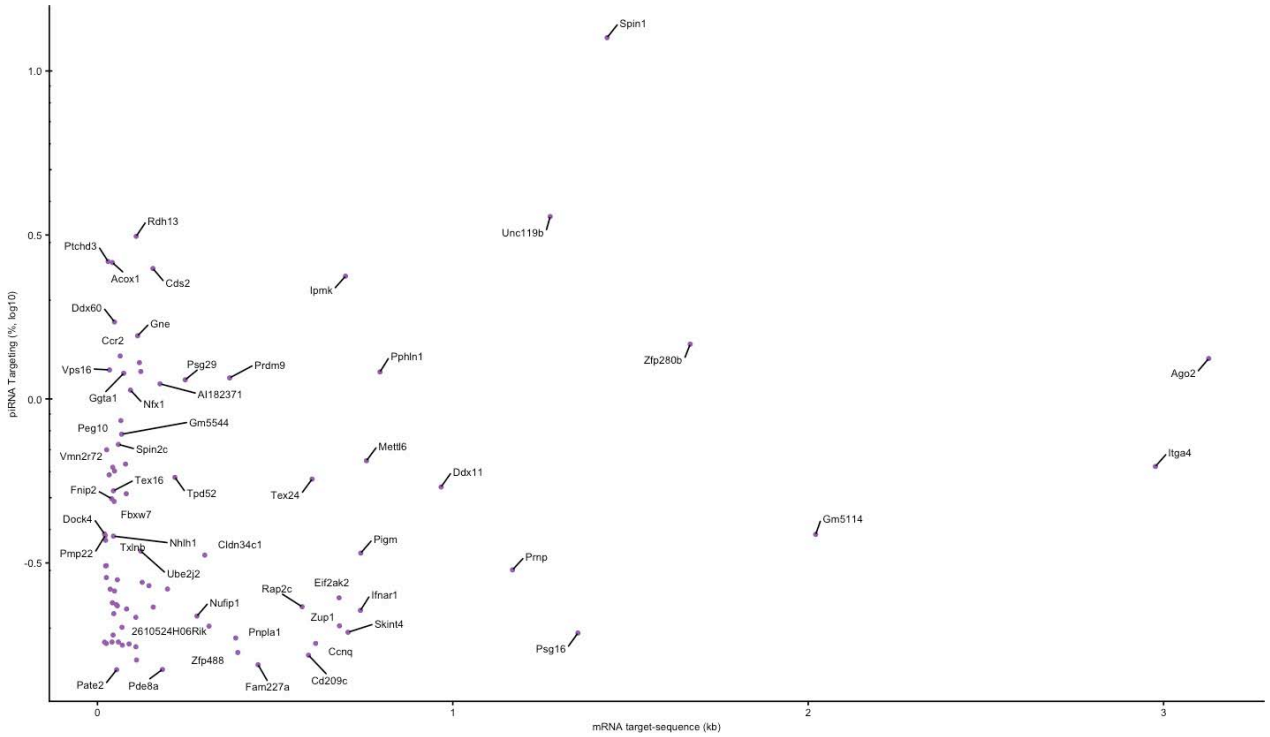
```
# Initialize PGasToTopPiC, check if gene is targeted by a piC that contains its
Pseudogene
targetingByPiRNAsSbO_sorted_total_tra$PGasToTopPiC <- FALSE
targetingByPiRNAsSbO_sorted_total_tra$PGasToTopPiC[targetingByPiRNAsSbO_sorted_total_tra
$topContribPiCrank == 0] <- NA

for (gene in targetingByPiRNAsSbO_sorted_total_tra$geneName) {
    if (gene %in% unique(unique_pairs$parent_id)) {
        assRank <- na.omit(unique_pairs[unique_pairs$parent_id == gene, "rank"])
        for (rank in assRank) {
            if
(targetingByPiRNAsSbO_sorted_total_tra[targetingByPiRNAsSbO_sorted_total_tra$geneName ==
gene,]$topContribPiCrank == rank) {

targetingByPiRNAsSbO_sorted_total_tra[targetingByPiRNAsSbO_sorted_total_tra$geneName ==
gene,]$PGasToTopPiC <- TRUE
            }
        }
    }
}
```

```
#get total coverage of targeting of gene

gr_alignments_red <- reduce(gr_alignments)

# Calculate overlap widths for matches
overlapsPiRNAsRedGenes <- findOverlaps(gr_alignments_red,
```

```
invertStrand(geneAnnotation_PCG_gene))
overlap_widths <-
tapply(width(pintersect(gr_alignments_red[queryHits(overlapsPiRNAsRedGenes)],

invertStrand(geneAnnotation_PCG_gene[subjectHits(overlapsPiRNAsRedGenes)]))),
                        subjectHits(overlapsPiRNAsRedGenes), sum)

# Create a vector of length equal to number of genes in geneAnnotation_PCG_gene
full_overlap_widths <- numeric(length(geneAnnotation_PCG_gene))

# Fill in the actual overlap values where they exist
full_overlap_widths[as.numeric(names(overlap_widths))] <- overlap_widths

# Create dataframe with all genes and their coverage
all_genes <- geneAnnotation_PCG_gene$gene_name
percentageCoverage_df <- data.frame(
    geneName = all_genes,
    targetingCoverage_bp = full_overlap_widths
)

# Merge with targeting dataframe
targetingByPiRNAsSbO_sorted_total_tra <- merge(
    targetingByPiRNAsSbO_sorted_total_tra,
    percentageCoverage_df,
    by = "geneName"
)

targetingByPiRNAsSbO_sorted_total_tra <-
targetingByPiRNAsSbO_sorted_total_tra[order(targetingByPiRNAsSbO_sorted_total_tra$totalP
iRNAcount, decreasing=TRUE),]
```

```
plot_targetingCoverage <-
ggplot(targetingByPiRNAsSbO_sorted_total_tra[1:genes_targetingThreshold,], aes(x =
targetingCoverage_bp/1000, y = log10(totalPiRNAperc))) +
  geom_point(color = "#9764aa", alpha = 1, size = 1) +
  geom_text_repel(aes(label = geneName),
                  size = 2.5,
                  box.padding = 0.5,
                  max.overlaps = 10) +
  annotation_logticks(base = 10, sides = "l", short = unit(0.02, "cm"), mid = unit(0.04,
"cm"), long = unit(0.06, "cm")) +
  theme_classic() +
  labs(
    x = "mRNA target-sequence (kb)",
    y = "piRNA Targeting (%, log10)"
  ) +
  theme(
    plot.title = element_text(size = 7, face = "bold"),
    axis.title = element_text(size = 7),
    axis.text = element_text(size = 7),
    legend.position = "none"
  )

plot_targetingCoverage
```

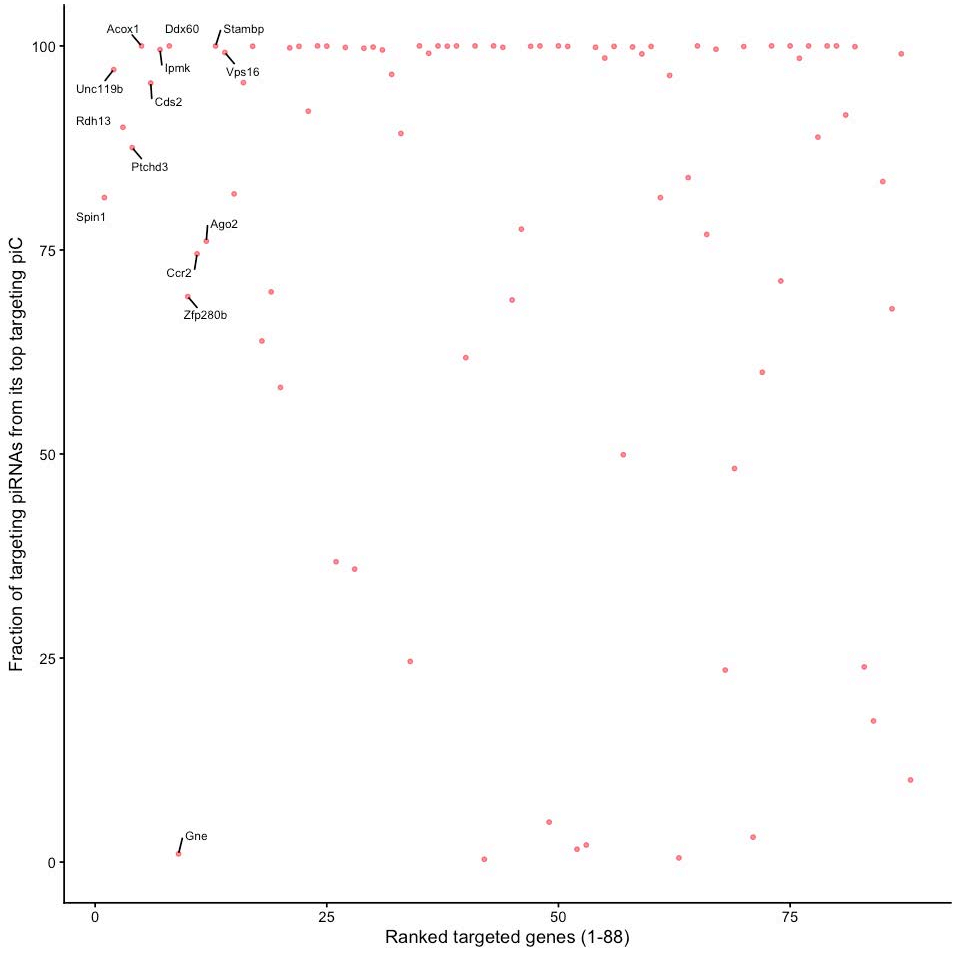
```
Warning message:
"ggrepel: 46 unlabeled data points (too many overlaps). Consider increasing max.
overlaps"
```



piRNA Cluster origins of targeting piRNAs (Fig. 1e, Extended Data Fig. 1d)

Fraction of piRNAs derived from their most targeting piRNA cluster

```
options(repr.plot.width=8, repr.plot.height=8)

# only label dots that rank 15 or higher in targeting
temp_targetingByPiRNAsSbO_sorted_total_tra <- targetingByPiRNAsSbO_sorted_total_tra %>%
  mutate(label = if_else(targetedRank < 15, geneName, NA_character_))

# plot fraction of targeting piRNAs from its top targeting piRNA cluster
plot_scTopPer <-
ggplot(temp_targetingByPiRNAsSbO_sorted_total_tra[1:genes_targetingThreshold,], aes(x =
targetedRank, y = top_piC_percentage*100)) +
    geom_point(color = "#ed1c24", alpha = 0.5, size = 1) +
    geom_text_repel(aes(label = label),
                    size = 2.5,
                    box.padding = 0.5,
                    max.overlaps = 10,
                    na.rm = TRUE) +
    scale_color_identity() +
    theme_classic() +
    labs(y = "Fraction of targeting piRNAs from its top targeting piC", x = "Ranked
targeted genes (1-88)") +
    ylim(0, 100)

plot_scTopPer
```



Fraction of gene-targeting piRNAs that originate from piRNA clusters that are not directly overlapping with the specific gene (in trans)

```
# put alignments to PCG transcriptome into context with the genes they target
# match read names, which include (among other things) information about the piC they
came from (rank) and their original read name (when mapped to mm10)
PCG_as_name_df <- as.data.frame(findOverlaps(gr_alignments,
invertStrand(geneAnnotation_PCG_gene)))
PCG_as_name_df$gene_name <-
geneAnnotation_PCG_gene[as.numeric(PCG_as_name_df$subjectHits), ]$gene_id
PCG_as_name_df$read_name <- sub("_.*", "",
gr_alignments[as.numeric(PCG_as_name_df$queryHits),]$qname)
PCG_as_name_df$read_nameWInfo <-
gr_alignments[as.numeric(PCG_as_name_df$queryHits),]$qname
PCG_as_name_df$rankOrigin <- sub("_|_", "", sub("rank|_", "",
regmatches(gr_alignments[as.numeric(PCG_as_name_df$queryHits),]$qname,
regexpr("rank(.*?)_", gr_alignments[as.numeric(PCG_as_name_df$queryHits),]$qname))))

# Precompute read–gene pairs from a single overlap against invertStrand(genes)
inv_genes <- invertStrand(genes)
hits <- findOverlaps(gr_mm10_prim, inv_genes)

# get read name - gene pairs that are antisense to eachother in mm10
mm10_as_pair <- unique(paste0(names(gr_mm10_prim)[as.integer(queryHits(hits))], "\r",
                              inv_genes$gene_id[as.integer(subjectHits(hits))]))

# Mark cis if (read_name, gene_name) observed in the precomputed pairs
```

```
PCG_as_pair <- paste0(PCG_as_name_df$read_name, "\r", PCG_as_name_df$gene_name)
PCG_as_name_df$cis_piRNA <- PCG_as_pair %in% mm10_as_pair
```

```
# all genes
# targeting in trans
nrow(PCG_as_name_df[!PCG_as_name_df$cis_piRNA,])/nrow(PCG_as_name_df)
# targeting in trans and from piRNA cluster
nrow(PCG_as_name_df[(!PCG_as_name_df$cis_piRNA) & (PCG_as_name_df$rankOrigin !=
0),])/nrow(PCG_as_name_df[!PCG_as_name_df$cis_piRNA,])
```

0.810230904999606

0.823757422229409

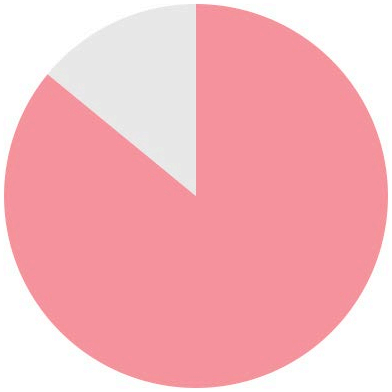
```
# top targeted genes (88)
PCG_as_name_df_topTargetedGenes <- PCG_as_name_df[PCG_as_name_df$gene_name %in%
targetingByPiRNAsSbO_sorted_total_tra$geneName[1:genes_targetingThreshold], ]

# targeting in trans
nrow(PCG_as_name_df_topTargetedGenes[!PCG_as_name_df_topTargetedGenes$cis_piRNA,])/nrow(
PCG_as_name_df_topTargetedGenes)
# targeting in trans and from piRNA cluster
nrow(PCG_as_name_df_topTargetedGenes[(!PCG_as_name_df_topTargetedGenes$cis_piRNA) &
(PCG_as_name_df_topTargetedGenes$rankOrigin !=
0),])/nrow(PCG_as_name_df_topTargetedGenes[!PCG_as_name_df_topTargetedGenes$cis_piRNA,])
```

0.964916661851023

0.859357521287267

```
# for top-targeted genes, fraction targeted by piRNAs from piRNA cluster (given that
they are trans targeting)
PCG_as_name_df_topTargetedGenes <- PCG_as_name_df[PCG_as_name_df$gene_name %in%
targetingByPiRNAsSbO_sorted_total_tra$geneName[1:genes_targetingThreshold], ]
transTargetingPiCpiRNAs <-
nrow(PCG_as_name_df_topTargetedGenes[(!PCG_as_name_df_topTargetedGenes$cis_piRNA) &
(PCG_as_name_df_topTargetedGenes$rankOrigin !=
0),])/nrow(PCG_as_name_df_topTargetedGenes[!PCG_as_name_df_topTargetedGenes$cis_piRNA,])
transTargetingPiCpiRNAs
# Create a data frame with this value
data <- data.frame(
  category = c("transTargetingPiCpiRNAs", "Other"),
  value = c(transTargetingPiCpiRNAs, 1-transTargetingPiCpiRNAs)
)

# Create the pie chart
options(repr.plot.width=4, repr.plot.height=4)
plot_byPiCTarg <- ggplot(data, aes(x = "", y = value, fill = category)) +
  geom_bar(stat = "identity", width = 1, alpha=0.5) +
  coord_polar(theta = "y") +
  scale_fill_manual(values = c("lightgrey", "#ed1c24")) +
  theme_void() +
  theme(legend.position = "none")

plot_byPiCTarg
```

0.859357521287267



Contribution of individual piRNA clusters to targeting (by their top, second, and third most contributing piC, other piCs and non-piC regions)

```
# filter custom PCG transcriptome coordinates of top targeted genes
subset_tra_gr <- geneAnnotation_PCG_gene[geneAnnotation_PCG_gene$gene_name %in%
targetingByPiRNAsSbO_sorted_total_tra[1:genes_targetingThreshold,]$geneName]
# add totalPiRNAcounts and targetedRank to GRange object
mcols(subset_tra_gr)$totalPiRNAcount <-
targetingByPiRNAsSbO_sorted_total_tra$totalPiRNAcount[match(mcols(subset_tra_gr)$gene_na
me, targetingByPiRNAsSbO_sorted_total_tra$geneName)]
mcols(subset_tra_gr)$targetedRank <-
targetingByPiRNAsSbO_sorted_total_tra$targetedRank[match(mcols(subset_tra_gr)$gene_name,
targetingByPiRNAsSbO_sorted_total_tra$geneName)]
```

```
# get table with piC-rank targeting contributions per gene
rank_piC_info <- function(overlaps) {
    if (length(overlaps) == 0) {
        return(c("0", 0, 0))
    }
    # Extract rank information using vectorized operations
    ranks <- sub("_|_", "", sub("rank|_", "", regmatches(overlaps$qname,
regexpr("rank(.*?)_", overlaps$qname))))
    rank_table <- table(ranks)

    return(rank_table)
}

# Get all targeting piRNAs in relation to the gene they target
overlap_hits <- findOverlaps(invertStrand(subset_tra_gr), gr_alignments)

# Split targeting piRNAs by gene
alignments_by_gene <- split(gr_alignments[subjectHits(overlap_hits)],
                            queryHits(overlap_hits))

# For each gene run rank_piC_info for table with piC-rank targeting contributions
piC_rank_summaries <- lapply(alignments_by_gene, rank_piC_info)
```

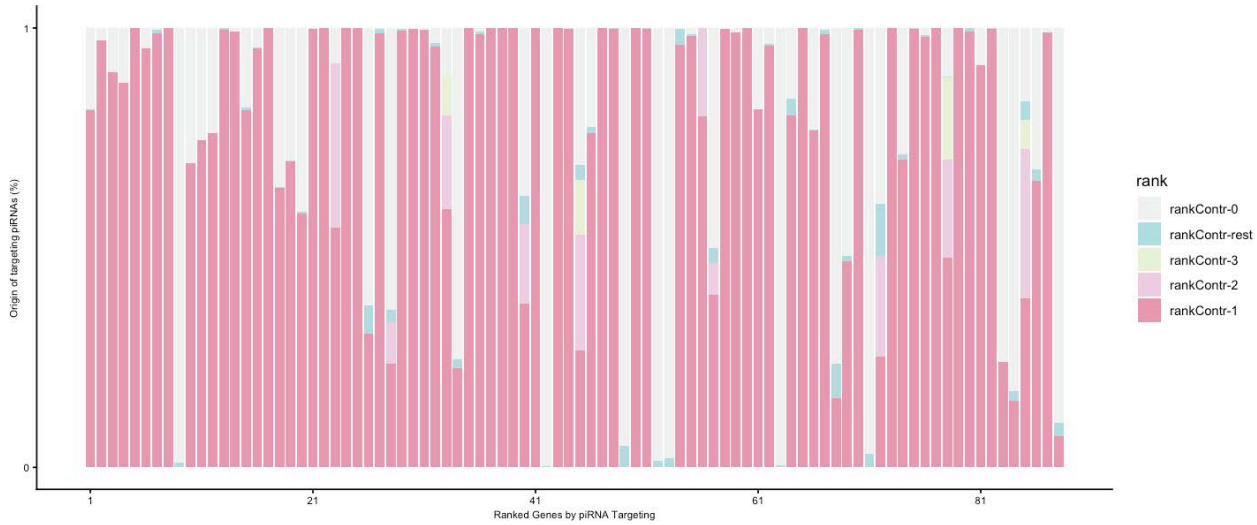
```
# Retrieve top contributing piC and percentage
# Separate piRNAs not from piCs and those from piCs that contribute < 5%
process_rank_contributions <- function(rank_table, total_count) {
    # Convert gene's piC_rank_summaries table to named vector and calc fractions
```

```
    contributions <- as.vector(rank_table) / total_count
    names(contributions) <- names(rank_table)

    # Separate category 0 (if it exists)
    cat_0 <- if("0" %in% names(contributions)) contributions["0"] else 0
    other_contributions <- contributions[names(contributions) != "0"]

    # Sort other contributions in descending order
    sorted_contributions <- sort(other_contributions, decreasing = TRUE)

    # Identify contributions >= 5%
    major_contributions <- sorted_contributions[sorted_contributions >= 0.05]
    minor_contributions <- sorted_contributions[sorted_contributions < 0.05]

    # Create result vector
    result <- c()
    result["rankContr-0"] <- cat_0

    # Add major contributions
    for(i in seq_along(major_contributions)) {
        result[paste0("rankContr-", i)] <- major_contributions[i]
    }

    # Sum minor contributions if any exist
    if(length(minor_contributions) > 0) {
        result["rankContr-rest"] <- sum(minor_contributions)
    }

    return(result)
}
```

```
# Apply to each gene and create new columns
contribution_results <- lapply(seq_along(alignments_by_gene), function(i) {
    rank_table <- piC_rank_summaries[[i]]
    total_count <- subset_tra_gr$totalPiRNAcount[i]
    process_rank_contributions(rank_table, total_count)
})
```

```
# Find all unique column names across all results
all_columns <- unique(unlist(lapply(contribution_results, names)))

# Ensure each result has all columns, filling missing ones with 0
contribution_results_normalized <- lapply(contribution_results, function(x) {
    missing_cols <- setdiff(all_columns, names(x))
    if(length(missing_cols) > 0) {
        x[missing_cols] <- 0
    }
    return(x[all_columns])
})
# convert to one data frame
contribution_df <- do.call(rbind, contribution_results_normalized)
colnames(contribution_df) <- all_columns

# Add contribution_df to the subset_tra_gr in df format
subset_tra <- cbind(as.data.frame(subset_tra_gr), contribution_df)
```

```
# Reshape to long format
data_long <- subset_tra %>%
  mutate(gene = rownames(subset_tra)) %>%
  gather(key = "rank", value = "value",
         starts_with("rankContr")) %>%
  mutate(rank = factor(rank,
                       levels = c("rankContr-0", "rankContr-rest", "rankContr-4",
"rankContr-3", "rankContr-2", "rankContr-1")))

# Create the stacked column chart
options(repr.plot.width=12, repr.plot.height=5)
plot_piCcontr <- ggplot(data_long, aes(x = targetedRank, y = value, fill = rank)) +
  geom_col(width = 0.85) +
  scale_y_continuous(breaks = c(0, 1), labels = c("0", "1")) +
    scale_x_continuous(breaks = seq(1, nrow(subset_tra), by = 20)) +
  scale_fill_manual(values = c("#F1F2F2", "#b3dee2", "#eaf2d7", "#efcfe3",
                               "#ea9ab2", "#e27396")) +
  theme_classic() +
  theme(
    axis.text = element_text(size = 7),
    axis.title = element_text(size = 7),
  ) +
  labs(x = "Ranked Genes by piRNA Targeting",
       y = "Origin of targeting piRNAs (%)")


plot_piCcontr
```



Targeting of gene features (Extended Data Fig. 1a)

```
# filter by top-targeted genes
geneAnnotation_PCG_gene_subset <-
geneAnnotation_PCG_gene[geneAnnotation_PCG_gene$gene_name %in%
targetingByPiRNAsSbO_sorted_total_tra$geneName[1:genes_targetingThreshold]]
selected_genes <- geneAnnotation_PCG_gene_subset$gene_name
```

```
unique(geneAnnotation_PCG$feature)
```

'gene' · '3UTR' · 'CDS' · '5UTR'

```
# subset geneAnnotation_PCG by genes (top targeted)
# and features (removing 'gene' annotation which includes all collapsed exons, not
seperated by features)
feature_list <- c("5UTR", "CDS", "3UTR")
gene_features <- geneAnnotation_PCG[geneAnnotation_PCG$gene_name %in% selected_genes &
geneAnnotation_PCG$feature %in% feature_list]
```

```
#pre-select alignments to only include alignments targeting selected genes
gr_alignments_main <- subsetByOverlaps(gr_alignments,
invertStrand(geneAnnotation_PCG_gene_subset))
mcols(gr_alignments_main) <- NULL
```

```
# make GRanges that have the starting position for each targeting piRNA
# For positive strand, make end = start
end(gr_alignments_main)[as.character(strand(gr_alignments_main)) == "+"] <-
start(gr_alignments_main)[as.character(strand(gr_alignments_main)) == "+"]

# For negative strand, make start = end
start(gr_alignments_main)[as.character(strand(gr_alignments_main)) == "-"] <-
end(gr_alignments_main)[as.character(strand(gr_alignments_main)) == "-"]
```

```
all_tiles <- NULL
# iterate through each gene
for (gene in unique(gene_features$gene_id)) {
    # get all piRNAs targeting that gene
    piRNA_startsAsToGene <- subsetByOverlaps(gr_alignments_main,
invertStrand(gene_features[gene_features$gene_id == gene]))

    # get gene coordinates in custom PCG transcriptome
    gene_ranges <- gene_features[gene_features$gene_id == gene]

    # iterate through each feature
    for (feature in feature_list) {

        # Subset to gene's feature
        temp_gr <- gene_ranges[gene_ranges$feature == feature]
        if (length(temp_gr) == 0) {
            # skip if gene does not have 5'UTR or 3'UTR
            next
        }

        # Merge overlapping or adjacent ranges
        merged_gr <- reduce(temp_gr)
        total_length <- sum(width(merged_gr))

        if (total_length < 20) {
            cat("   Feature ", feature, " for gene ", gene, " too short (<20 nt total).
Skipping.\n")
            next
        }

        # Figure out which direction to tile
        gene_strand <- unique(as.character(strand(temp_gr)))
        merged_gr <- sort(merged_gr)
```

```
# Determine exact tile sizes so that each tile is ~5%
base_tile_size <- floor(total_length / 20)
leftover <- total_length %% 20
tile_sizes <- rep(base_tile_size, 20)

# Distribute the remainder (leftover) among the first tiles
if (leftover > 0) {
    if (gene_strand == "+") {
        tile_sizes[seq_len(leftover)] <- tile_sizes[seq_len(leftover)] + 1
    } else {
        tile_sizes[21-seq_len(leftover)] <- tile_sizes[21-seq_len(leftover)] + 1
    }
}

# Build the 20 tiles by walking through merged_gr
tile_list <- vector("list", 20)
current_tile_index <- 1
target_tile_len <- tile_sizes[current_tile_index]
cum_len_in_tile    <- 0
current_ranges     <- IRanges()

# Helper to finalize a tile and reset
finalize_tile <- function() {
    tile_list[[current_tile_index]] <<- GRanges(
        seqnames = seqnames(merged_gr)[1],
        ranges   = current_ranges,
        strand   = gene_strand,
        gene_id  = gene,
        feature  = feature,
        tile_index = current_tile_index
    )

    current_tile_index <<- current_tile_index + 1
    if (current_tile_index <= 20) {
        target_tile_len <<- tile_sizes[current_tile_index]
    }
    cum_len_in_tile <<- 0
    current_ranges <<- IRanges()
}

for (seg in seq_along(merged_gr)) {
    seg_start <- start(merged_gr[seg])
    seg_end   <- end(merged_gr[seg])
    seg_width <- width(merged_gr[seg])

    bases_used_in_seg <- 0

    # Iterate base by base in principle, but slice big chunks if possible
    while (bases_used_in_seg < seg_width && current_tile_index <= 20) {

        # Still need 'remaining_in_tile' bases to complete the current tile
        needed_for_tile <- target_tile_len - cum_len_in_tile
        # The maximum we can take from the current segment is what's left in it
        left_in_segment <- seg_width - bases_used_in_seg
        # The actual chunk we'll consume from this segment
        chunk_size <- min(needed_for_tile, left_in_segment)

        if (chunk_size == 0) {
            # tile is exactly filled
```

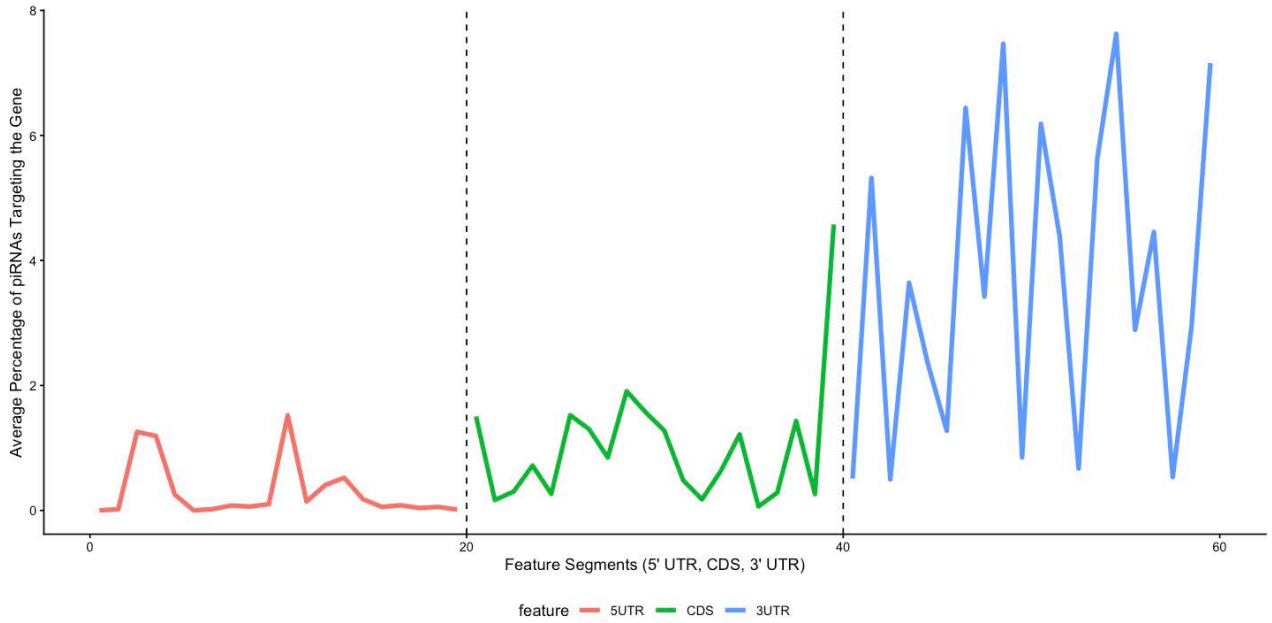
```
                    finalize_tile()
                    if (current_tile_index > 20) break
                        next
                }


                chunk_start <- seg_start + bases_used_in_seg
                chunk_end   <- chunk_start + chunk_size - 1


                # Add IRanges chunk
                current_ranges <- c(
                current_ranges,
                IRanges(start = chunk_start, end = chunk_end)
                )

                # Update counters
                bases_used_in_seg <- bases_used_in_seg + chunk_size
                cum_len_in_tile    <- cum_len_in_tile + chunk_size

                # If tile is filled, finalize
                if (cum_len_in_tile == target_tile_len) {
                    finalize_tile()
                    if (current_tile_index > 20) break
                }
            }
            if (current_tile_index > 20) break
        }

        # If something left in the last tile
        if (current_tile_index <= 20 && cum_len_in_tile > 0) {
            finalize_tile()
        }

        final_tiles <- do.call(c, tile_list)  # a GRanges of length 20

        #handle minus strand by just reversing the column tile_index (20 to 1, 19 to 2,
etc)
        if (gene_strand == "-") {
            final_tiles$tile_index <- 20 - final_tiles$tile_index + 1
        }

        # Count overlaps for each tile and add these counts to metacolumn of final_tiles
        overlap_counts <- countOverlaps(final_tiles, invertStrand(gr_alignments_main))
        mcols(final_tiles)$read_counts <- overlap_counts

        #combine to all_tiles
        all_tiles <- c(all_tiles, final_tiles)

  }
}
all_tiles <- do.call(c, all_tiles)
```

```
            Feature  5UTR  for gene  Zc3hav1l  too short (<20 nt total). Skipping.
            Feature  5UTR  for gene  Rdh8  too short (<20 nt total). Skipping.
            Feature  5UTR  for gene  Tex16  too short (<20 nt total). Skipping.
```

```
# Calc percentages of targeting piRNAs for each gene
all_tiles_df <- as.data.frame(all_tiles) %>%
  group_by(gene_id) %>%
```

```
  mutate(total_counts = sum(read_counts),
         percentage = (read_counts / total_counts) * 100)

all_tiles_df$feature <- factor(all_tiles_df$feature, levels = c("5UTR", "CDS", "3UTR"))
```

```
options(repr.plot.width=12, repr.plot.height=6)

# to plot them next to each other
all_tiles_df <- all_tiles_df %>%
  mutate(
    Adjusted_Bin = case_when(
      feature == "5UTR" ~ tile_index - 0.5,
      feature == "CDS" ~ tile_index + 19.5,
      feature == "3UTR" ~ tile_index + 39.5
    )
  )

# Calculate averages for the average plot
all_tiles_df_avg <- all_tiles_df %>%
  group_by(feature, Adjusted_Bin) %>%
  summarise(avg_percentage = mean(percentage, na.rm = TRUE), .groups = "drop")

# Averaged piRNA targeting across all genes with segmented x-axis
featureTargetingPlotTotal <- ggplot(all_tiles_df_avg, aes(x = Adjusted_Bin, y =
avg_percentage, color = feature)) +
  geom_line(linewidth = 1.5) +
  geom_vline(xintercept = c(20, 40), linetype = "dashed", color = "black") +
  labs(
    x = "Feature Segments (5' UTR, CDS, 3' UTR)",
    y = "Average Percentage of piRNAs Targeting the Gene"
  ) +
  theme_classic() +
  theme(
    legend.position = "bottom"
  )
featureTargetingPlotTotal
```

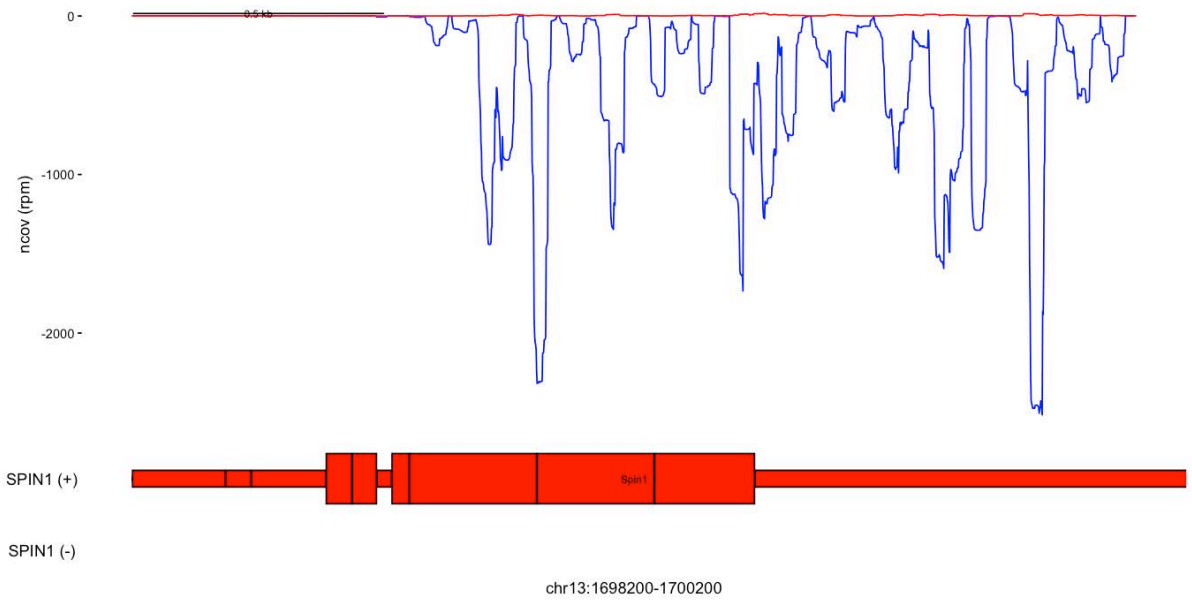


Coverage Plots (Fig. 1b, Extended Data Fig. 1b,e,f)

Gene-targeting (shown in Custom Protein Coding Gene Transcriptome)

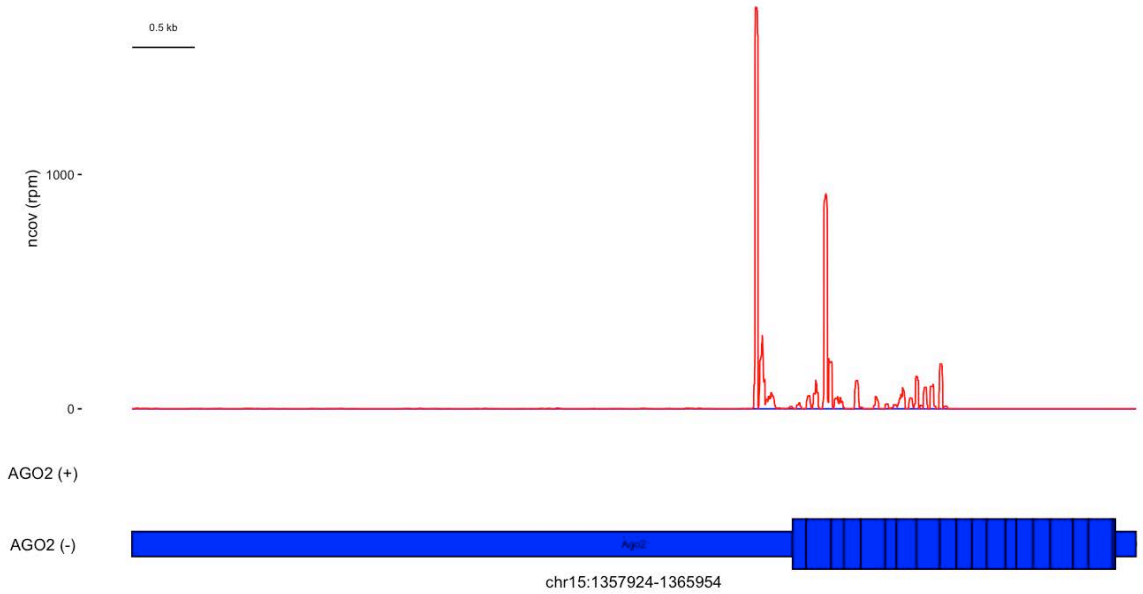
```
# SPIN1 - Figure 1b
options(repr.plot.width=12, repr.plot.height=6)
chr <- "chr13"
start <- 1698200
end <- 1700200
coord <- IRanges(start, end)
geneAnnotation_PCG$type <- geneAnnotation_PCG$feature

SPIN1targeting<- allTracksPlotted(piRNAs_from_Bam = gr_alignments, chromosome = chr,
IRangesCoord=coord, gtfFiles=list(SPIN1 = geneAnnotation_PCG), tilesWidth=1,
scaleWidthKB = 0.5)
SPIN1targetingFull <- SPIN1targeting$plotCoverageTrack /
SPIN1targeting$trackAll$gtfNum1$trackPlus / SPIN1targeting$trackAll$gtfNum1$trackMinus
SPIN1targetingFull
```

```
Normalizing to RPM
```




```
# AGO2 - Fig Extended Data 1b
chr <- "chr15"
start <- 1357924
end <- 1365954
coord <- IRanges(start, end)
geneAnnotation_PCG$type <- geneAnnotation_PCG$feature

# showing only targeting piRNAs (disregard therefore normalization, not shown in
manuscript)
AGO2targeting<- allTracksPlotted(piRNAs_from_Bam = gr_alignments[strand(gr_alignments)
== "+"], chromosome = chr, IRangesCoord=coord, gtfFiles=list(AGO2 = geneAnnotation_PCG),
tilesWidth=1, scaleWidthKB = 0.5)
AGO2targetingFull <- AGO2targeting$plotCoverageTrack /
AGO2targeting$trackAll$gtfNum1$trackPlus / AGO2targeting$trackAll$gtfNum1$trackMinus
AGO2targetingFull
```

```
Normalizing to RPM
```



Gene-targeting (shown in mm10 genome)

Retrieve and save genomic sequence of Spin1 and Ago2

```
# load gene coordinates in mm10 genome, uncollapsed genes
genesRefSeq_dir <- "../data/annotations/mm10.ncbiRefSeq.gtf"
genesRefSeq <- rtracklayer::import(genesRefSeq_dir)
```

```
# Extract Spin1 gene annotations, excluding predicted annotations
geneAnnotation_PCG_Spin1 <- genesRefSeq[genesRefSeq$gene_id == "Spin1",]
geneAnnotation_PCG_Spin1 <- geneAnnotation_PCG_Spin1[!grepl("^X",
geneAnnotation_PCG_Spin1$transcript_id)]

# Spin1 region in mm10 genome
chr <- seqnames(geneAnnotation_PCG_Spin1[1])
start <- min(start(geneAnnotation_PCG_Spin1)) - 50
end <- max(end(geneAnnotation_PCG_Spin1)) + 50
message(start, "-", end)
# Get sequence and create fasta
genome <- BSgenome.Mmusculus.UCSC.mm10
sequence <- DNAStringSet(getSeq(genome, chr, start, end))
names(sequence) <- chr

writeXStringSet(sequence,
                filepath=paste0("../data/others/FM_Spin1region_mm10/regionSpin1_", chr,
"_", start, "_", end, ".fa"))
```

```
51100830-51152612
```

```
# Extract Ago2 gene annotations, excluding predicted annotations
geneAnnotation_PCG_Ago2 <- genesRefSeq[genesRefSeq$gene_id == "Ago2",]
geneAnnotation_PCG_Ago2 <- geneAnnotation_PCG_Ago2[!grepl("^X",
geneAnnotation_PCG_Ago2$transcript_id)]

# Ago2 region in mm10 genome
chr <- seqnames(geneAnnotation_PCG_Ago2[1])
```

```
start <- min(start(geneAnnotation_PCG_Ago2)) - 50
end <- max(end(geneAnnotation_PCG_Ago2)) + 50

# Get sequence and create fasta
genome <- BSgenome.Mmusculus.UCSC.mm10
sequence <- DNAStringSet(getSeq(genome, chr, start, end))
names(sequence) <- chr

writeXStringSet(sequence,
                filepath=paste0("../data/others/FM_Ago2region_mm10/regionAgo2_", chr,
"_", start, "_", end, ".fa"))
```

Map to genomic sequence of Spin1 and Ago2

BASH

```
# Index the FASTA file for STAR
STAR --runMode genomeGenerate \
     --genomeDir
../data/others/FM_Spin1region_mm10/regionSpin1_chr13_51100830_51152612_dir \
     --genomeFastaFiles
../data/others/FM_Spin1region_mm10/regionSpin1_chr13_51100830_51152612.fa \
     --genomeSAindexNbases 7
```

```
        Oct 30 12:57:24 ..... started STAR run
        Oct 30 12:57:24 ... starting to generate Genome files
        Oct 30 12:57:24 ... starting to sort Suffix Array. This may take a long time...
        Oct 30 12:57:24 ... sorting Suffix Array chunks and saving them to disk...
        Oct 30 12:57:24 ... loading chunks from disk, packing SA...
        Oct 30 12:57:24 ... finished generating suffix array
        Oct 30 12:57:24 ... generating Suffix Array index
        Oct 30 12:57:24 ... completed Suffix Array index
        Oct 30 12:57:24 ... writing Genome to disk ...
        Oct 30 12:57:24 ... writing Suffix Array to disk ...
        Oct 30 12:57:24 ... writing SAindex to disk
        Oct 30 12:57:24 ..... finished successfully
```

```
#bash
input_fasta="../data/annotations/customTranscriptomes/Mouse_161922_testes_small_RNAs_ZL6
_S7_R1_trimmed_UMIcollapsed_structOut_miRoutwS_Aligned.PICBloadWseqs.primAlignWinfo.fast
a"

addFastaChange=""
addMappingChange="clip5pNbases1_Extend5pOfRead1_minMatch19"

#minimum Match of 19 nt, to restrict soft-clipping
STAR --runMode alignReads \
    --runThreadN 20 \
    --genomeDir
../data/others/FM_Spin1region_mm10/regionSpin1_chr13_51100830_51152612_dir \
    --readFilesIn $input_fasta \
    --clip5pNbases 1 \
    --alignEndsType Extend5pOfRead1 \
    --outSAMattributes All \
    --outSAMtype BAM SortedByCoordinate \
    --limitBAMsortRAM 20000000000 \
    --alignIntronMax 1 \
    --alignSoftClipAtReferenceEnds No \
    --outFilterMismatchNmax 1 \
    --outFilterMatchNmin 19 \
```

```
    --winAnchorMultimapNmax 100 \
    --outFilterMultimapNmax 100 \
    --outReadsUnmapped Fastx \
    --outFileNamePrefix
../data/others/FM_Spin1region_mm10/Spin1RegionOnlyWIntrons_PCG_Mouse_161922.Aligned.PICB
loadWseqs.primAlignWinfo.${addFastaChange}.Spin1_chr13_51100830_51152612_${addMappingCha
nge}_

echo "Mapped to Spin1 region (in mm10 genome)"
```

```
        Oct 30 12:57:35 ..... started STAR run
        Oct 30 12:57:35 ..... loading genome
        Oct 30 12:57:35 ..... started mapping
        Oct 30 13:00:19 ..... started sorting BAM
        Oct 30 13:00:19 ..... finished successfully
        Mapped to Spin1 region (in mm10 genome)
```

```
# Index the FASTA file for STAR
STAR --runMode genomeGenerate \
     --genomeDir
../data/others/FM_Ago2region_mm10/regionAgo2_chr15_73101575_73184997_dir \
     --genomeFastaFiles
../data/others/FM_Ago2region_mm10/regionAgo2_chr15_73101575_73184997.fa \
     --genomeSAindexNbases 7
```

```
        Oct 30 11:28:18 ..... started STAR run
        Oct 30 11:28:18 ... starting to generate Genome files
        Oct 30 11:28:18 ... starting to sort Suffix Array. This may take a long time...
        Oct 30 11:28:18 ... sorting Suffix Array chunks and saving them to disk...
        Oct 30 11:28:18 ... loading chunks from disk, packing SA...
        Oct 30 11:28:18 ... finished generating suffix array
        Oct 30 11:28:18 ... generating Suffix Array index
        Oct 30 11:28:18 ... completed Suffix Array index
        Oct 30 11:28:18 ... writing Genome to disk ...
        Oct 30 11:28:18 ... writing Suffix Array to disk ...
        Oct 30 11:28:18 ... writing SAindex to disk
        Oct 30 11:28:18 ..... finished successfully
```

```
#bash
input_fasta="../data/annotations/customTranscriptomes/Mouse_161922_testes_small_RNAs_ZL6
_S7_R1_trimmed_UMIcollapsed_structOut_miRoutwS_Aligned.PICBloadWseqs.primAlignWinfo.fast
a"

addFastaChange=""
addMappingChange="clip5pNbases1_Extend5pOfRead1_minMatch19"

#minimum Match of 19 nt, to restrict soft-clipping
STAR --runMode alignReads \
    --runThreadN 20 \
    --genomeDir ../data/others/FM_Ago2region_mm10/regionAgo2_chr15_73101575_73184997_dir
\
    --readFilesIn $input_fasta \
    --clip5pNbases 1 \
    --alignEndsType Extend5pOfRead1 \
    --outSAMattributes All \
    --outSAMtype BAM SortedByCoordinate \
    --limitBAMsortRAM 20000000000 \
    --alignIntronMax 1 \
    --alignSoftClipAtReferenceEnds No \
    --outFilterMismatchNmax 1 \
```

```
    --outFilterMatchNmin 19 \
    --winAnchorMultimapNmax 100 \
    --outFilterMultimapNmax 100 \
    --outReadsUnmapped Fastx \
    --outFileNamePrefix
../data/others/FM_Ago2region_mm10/Ago2RegionOnlyWIntrons_PCG_Mouse_161922_Aligned.PICBlo
adWseqs.primAlignWinfo.${addFastaChange}.Ago2_chr15_73101575_73184997_${addMappingChange
}_

echo "Mapped to Ago2 region (in mm10 genome)"
```

```
        Oct 30 11:39:07 ..... started STAR run
        Oct 30 11:39:08 ..... loading genome
        Oct 30 11:39:08 ..... started mapping
        Oct 30 11:42:18 ..... started sorting BAM
        Oct 30 11:42:18 ..... finished successfully
        Mapped to Ago2 region (in mm10 genome)
```

Build coverage plots for alignments in Spin1 and Ago2 genomic regions (Fig S1e and S1f)

R - Rerun 'Prep'-section of code first

```
# load gene coordinates in mm10 genome, uncollapsed genes
genesRefSeq_dir <- "../data/annotations/mm10.ncbiRefSeq.gtf"
genesRefSeq <- rtracklayer::import(genesRefSeq_dir)
```

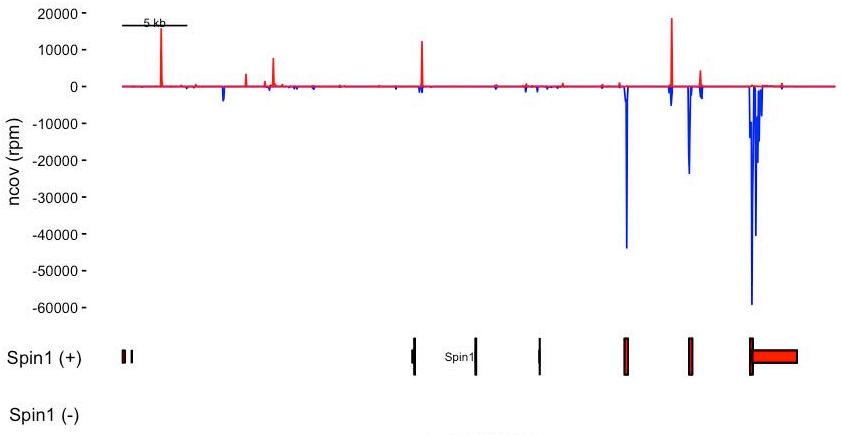
```
# Load bam file and keep only unique alignments that map uniquely both to Spin1 and also
mm10
bamSpin1_dir <-
"../data/others/FM_Spin1region_mm10/Spin1RegionOnlyWIntrons_PCG_Mouse_161922.Aligned.PIC
BloadWseqs.primAlignWinfo..Spin1_chr13_51100830_51152612_clip5pNbases1_Extend5pOfRead1_m
inMatch19_Aligned.sortedByCoord.out.bam"
bam <- BamFile(bamSpin1_dir)

fields <- scanBamWhat()
primary_flag <- scanBamFlag(isSecondary = FALSE, isSupplementary = FALSE)
param <- ScanBamParam(flag = primary_flag,
    what=c('qname','flag','rname','strand','pos','qwidth', 'cigar', 'seq'),
    tag=c('NH'))

ga_Spin1_alignments <- readGAlignments(bam, param = param)
ga_Spin1_alignments <- ga_Spin1_alignments[mcols(ga_Spin1_alignments)$NH == 1]

gr_Spin1_alignments <- makeGRangesFromDataFrame(as.data.frame(ga_Spin1_alignments),
keep.extra.columns=TRUE)

mcols(gr_Spin1_alignments) <- mcols(gr_Spin1_alignments)[ , !
(colnames(mcols(gr_Spin1_alignments)) %in% c("njunc", "strandInfo", "rname", "strand.1",
"pos", "qwidth.1", "cigar.1", "qual"))]

# consider only reads mapped uniquely to mm10
gr_Spin1_alignments <- gr_Spin1_alignments[grepl("_NH1$",
mcols(gr_Spin1_alignments)$qname)]
```

```
# get Spin1 info, exclude predicted annotations and shift to match Spin1 transcriptome
geneAnnotation_PCG_Spin1 <- genesRefSeq[genesRefSeq$gene_id == "Spin1",]
geneAnnotation_PCG_Spin1 <- geneAnnotation_PCG_Spin1[!grepl("^X",
geneAnnotation_PCG_Spin1$transcript_id)]
```

```
geneAnnotation_PCG_Spin1_shift <- GenomicRanges::shift(geneAnnotation_PCG_Spin1, shift =
-(min(start(geneAnnotation_PCG_Spin1)) - 50))
```

```
# Create coverage plot with custom function allTracksPlotted
chr <- as.character(seqnames(genes[genes$gene_id == "Spin1"][1]))
start <- 1
end <- (max(end(genes[genes$gene_id == "Spin1"])) + 50) - (min(start(genes[genes$gene_id
== "Spin1"])) - 50)
coord <- IRanges(start, end)

# Spin1 annotations in region
geneAnnotation_PCG_Spin1_shift <- subsetByOverlaps(
    geneAnnotation_PCG_Spin1_shift[geneAnnotation_PCG_Spin1_shift$type %in% c("CDS",
"5UTR", "3UTR")],
    GRanges(seqnames = chr, ranges = coord))

# Build coverage track of full region
options(repr.plot.width=8, repr.plot.height=5)
Spin1targetingWintrons <- allTracksPlotted(piRNAs_from_Bam = gr_Spin1_alignments,
chromosome = chr, IRangesCoord=coord, gtfFiles=list(Spin1 =
geneAnnotation_PCG_Spin1_shift), tilesWidth=50, scaleWidthKB = 5)
Spin1targetingWintronsFull <- Spin1targetingWintrons$plotCoverageTrack /
Spin1targetingWintrons$trackAll$gtfNum1$trackPlus /
Spin1targetingWintrons$trackAll$gtfNum1$trackMinus
Spin1targetingWintronsFull
```

```
Normalizing to RPM
```



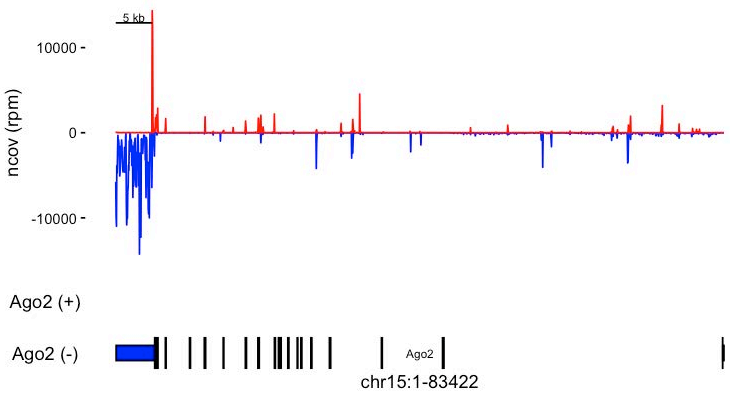

```
# Load bam file and keep only unique alignments that map uniquely both to Ago2 and also
mm10
bamAgo2_dir <-
"../data/others/FM_Ago2region_mm10/Ago2RegionOnlyWIntrons_PCG_Mouse_161922_Aligned.PICBl
oadWseqs.primAlignWinfo..Ago2_chr15_73101575_73184997_clip5pNbases1_Extend5pOfRead1_minM
atch19_Aligned.sortedByCoord.out.bam"
bam <- BamFile(bamAgo2_dir)

fields <- scanBamWhat()
primary_flag <- scanBamFlag(isSecondary = FALSE, isSupplementary = FALSE)
param <- ScanBamParam(flag = primary_flag,
```

```
    what=c('qname','flag','rname','strand','pos','qwidth', 'cigar', 'seq'),
    tag=c('NH'))

ga_Ago2_alignments <- readGAlignments(bam, param = param)
ga_Ago2_alignments <- ga_Ago2_alignments[mcols(ga_Ago2_alignments)$NH == 1]

gr_Ago2_alignments <- makeGRangesFromDataFrame(as.data.frame(ga_Ago2_alignments),
keep.extra.columns=TRUE)

mcols(gr_Ago2_alignments) <- mcols(gr_Ago2_alignments)[ , !
(colnames(mcols(gr_Ago2_alignments)) %in% c("njunc", "strandInfo", "rname", "strand.1",
"pos", "qwidth.1", "cigar.1", "qual"))]

# consider only reads mapped uniquely to mm10
gr_Ago2_alignments <- gr_Ago2_alignments[grepl("_NH1$",
mcols(gr_Ago2_alignments)$qname)]
```

```
# get Ago2 info, exclude predicted annotations and shift to match Ago2 transcriptome
geneAnnotation_PCG_Ago2 <- genesRefSeq[genesRefSeq$gene_id == "Ago2",]
geneAnnotation_PCG_Ago2 <- geneAnnotation_PCG_Ago2[!grepl("^X",
geneAnnotation_PCG_Ago2$transcript_id)]
geneAnnotation_PCG_Ago2_shift <- GenomicRanges::shift(geneAnnotation_PCG_Ago2, shift = -
(min(start(geneAnnotation_PCG_Ago2)) - 50))
```

```
# Create coverage plot with custom function allTracksPlotted
chr <- as.character(seqnames(genes[genes$gene_id == "Ago2"][1]))
start <- 1
end <- (max(end(geneAnnotation_PCG_Ago2)) + 50) - (min(start(geneAnnotation_PCG_Ago2)) -
50)
coord <- IRanges(start, end)

# Ago2 annotations in region
geneAnnotation_PCG_Ago2_shift <- subsetByOverlaps(
    geneAnnotation_PCG_Ago2_shift[geneAnnotation_PCG_Ago2_shift$type %in% c("CDS",
"5UTR", "3UTR")],
    GRanges(seqnames = chr, ranges = coord))

# Build coverage track of full region
Ago2targetingWintrons <- allTracksPlotted(piRNAs_from_Bam = gr_Ago2_alignments,
chromosome = chr, IRangesCoord=coord, gtfFiles=list(Ago2 =
geneAnnotation_PCG_Ago2_shift), tilesWidth=50, scaleWidthKB = 5)
Ago2targetingWintronsFull <- Ago2targetingWintrons$plotCoverageTrack /
Ago2targetingWintrons$trackAll$gtfNum1$trackPlus /
Ago2targetingWintrons$trackAll$gtfNum1$trackMinus
Ago2targetingWintronsFull
```

```
        Normalizing to RPM
```



piRNA Cluster (piC) region as the origin of gene targeting

Spin1 (Fig 1f)

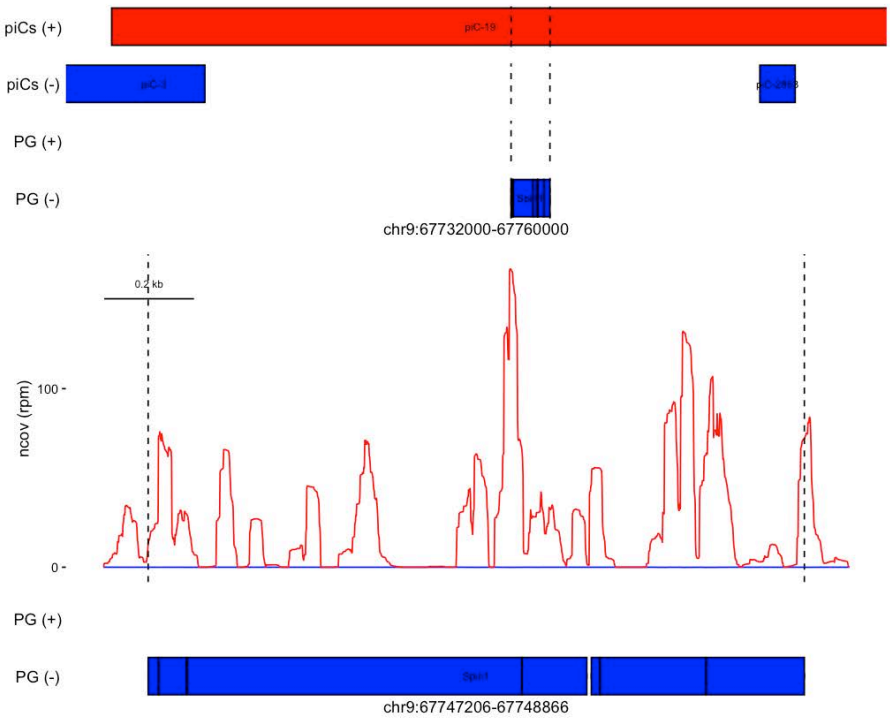
```
# Spin1 - full gene region
message("Full piC-as(Spin1) region: ", as.character(seqnames(MILIclusters_pach[19])[1]),
":", start(MILIclusters_pach[19]), "-", end(MILIclusters_pach[19]))
# slight zoom in to focus on Pseudogene-region
chr <- seqnames(MILIclusters_pach[19])[1]
start <- 67732000
end <- 67760000
coord <- IRanges(start, end)
PG_Spin1 <- subsetByOverlaps(pseudogenesRGOvrlp, invertStrand(MILIclusters_pach[19]))

PG_Spin1$gene_name <- PG_Spin1$parent_id
MILIclusters_pach$gene_name <- paste0("rank_",
MILIclusters_pach$rankByAllReadsPrimaryAlignmentsFPM)

Spin1originPiCzoomOut <- allTracksPlotted(chromosome = chr, IRangesCoord=coord,
gtfFiles=list(piCs = MILIclusters_pach, PG = PG_Spin1), tilesWidth=50, scaleWidthKB = 1)

Spin1originPiCzoomOutFull <- Spin1originPiCzoomOut$plotCoverageTrack +
        geom_vline(xintercept = min(start(PG_Spin1[PG_Spin1$parent_id == "Spin1"])),
linetype = "dashed", color = "black") +
        geom_vline(xintercept = max(end(PG_Spin1[PG_Spin1$parent_id == "Spin1"])),
linetype = "dashed", color = "black")

t1p <- Spin1originPiCzoomOut$trackAll$gtfNum1$trackPlus  +
        geom_vline(xintercept = min(start(PG_Spin1[PG_Spin1$parent_id == "Spin1"])),
linetype = "dashed", color = "black") +
        geom_vline(xintercept = max(end(PG_Spin1[PG_Spin1$parent_id == "Spin1"])),
linetype = "dashed", color = "black")
```

```
t1m <- Spin1originPiCzoomOut$trackAll$gtfNum1$trackMinus  +
        geom_vline(xintercept = min(start(PG_Spin1[PG_Spin1$parent_id == "Spin1"])),
linetype = "dashed", color = "black") +
        geom_vline(xintercept = max(end(PG_Spin1[PG_Spin1$parent_id == "Spin1"])),
linetype = "dashed", color = "black")

t2p <- Spin1originPiCzoomOut$trackAll$gtfNum2$trackPlus  +
        geom_vline(xintercept = min(start(PG_Spin1[PG_Spin1$parent_id == "Spin1"])),
linetype = "dashed", color = "black") +
        geom_vline(xintercept = max(end(PG_Spin1[PG_Spin1$parent_id == "Spin1"])),
linetype = "dashed", color = "black")

t2m <- Spin1originPiCzoomOut$trackAll$gtfNum2$trackMinus  +
        geom_vline(xintercept = min(start(PG_Spin1[PG_Spin1$parent_id == "Spin1"])),
linetype = "dashed", color = "black") +
        geom_vline(xintercept = max(end(PG_Spin1[PG_Spin1$parent_id == "Spin1"])),
linetype = "dashed", color = "black")
```

```
Full piC-as(Spin1) region: chr9:67732316-67772950
```

```
# Spin1 zoom in
message("Spin1-PG region: ", as.character(seqnames(PG_Spin1[PG_Spin1$parent_id ==
"Spin1"])[1]), ":", min(start(PG_Spin1[PG_Spin1$parent_id == "Spin1"])), "-",
max(end(PG_Spin1[PG_Spin1$parent_id == "Spin1"])))
chr <- "chr9"
start <- min(start(PG_Spin1[PG_Spin1$parent_id == "Spin1"])) - 100
end <- max(end(PG_Spin1[PG_Spin1$parent_id == "Spin1"])) + 100
coord <- IRanges(start, end)

# add dashed lines to plot to indicate pseudogene region
Spin1originPiCzoomIn <- allTracksPlotted(piRNAs_from_Bam = gr_mm10, chromosome = chr,
IRangesCoord=coord, gtfFiles=list(PG = PG_Spin1), tilesWidth=1, scaleWidthKB = 0.2)
Spin1originPiCzoomInFull <- Spin1originPiCzoomIn$plotCoverageTrack  +
        geom_vline(xintercept = min(start(PG_Spin1[PG_Spin1$parent_id == "Spin1"])),
linetype = "dashed", color = "black") +
        geom_vline(xintercept = max(end(PG_Spin1[PG_Spin1$parent_id == "Spin1"])),
linetype = "dashed", color = "black")

Spin1originPiCFull <- t1p / t1m / t2p / t2m / Spin1originPiCzoomInFull /
    Spin1originPiCzoomIn$trackAll$gtfNum1$trackPlus /
Spin1originPiCzoomIn$trackAll$gtfNum1$trackMinus /
    Spin1originPiCzoomIn$trackAll$gtfNum2$trackPlus /
Spin1originPiCzoomIn$trackAll$gtfNum2$trackMinus

Spin1originPiCFull
```

```
Spin1-PG region: chr9:67747306-67748766

Normalizing to RPM
```



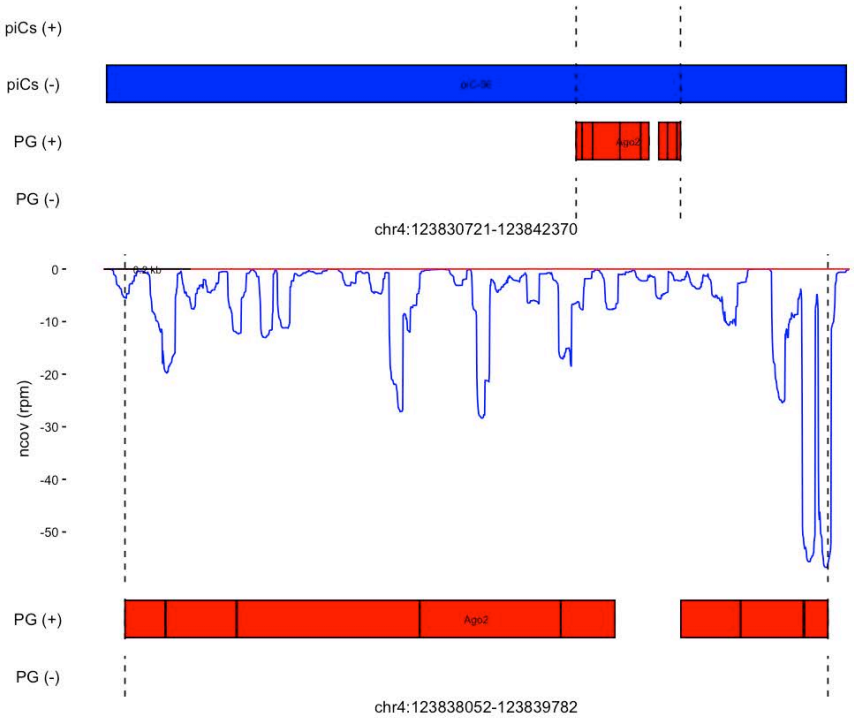
Ago2 (Fig 1g)

```
# Ago2 - full gene region
chr <- seqnames(MILIclusters_pach[56])[1]
start <- start(MILIclusters_pach[56]) - 50
end <- end(MILIclusters_pach[56]) + 50
coord <- IRanges(start, end)
PG_Ago2 <- subsetByOverlaps(pseudogenesRGOvrlp, invertStrand(MILIclusters_pach[56]))

PG_Ago2$gene_name <- PG_Ago2$parent_id
MILIclusters_pach$gene_name <- paste0("rank_",
MILIclusters_pach$rankByAllReadsPrimaryAlignmentsFPM)

Ago2originPiCzoomOut <- allTracksPlotted(chromosome = chr, IRangesCoord=coord,
gtfFiles=list(piCs = MILIclusters_pach[56], PG = PG_Ago2[PG_Ago2$parent_id == "Ago2"]),
tilesWidth=50, scaleWidthKB = 1)
Ago2originPiCzoomOutFull <- Ago2originPiCzoomOut$plotCoverageTrack  +
        geom_vline(xintercept = min(start(PG_Ago2[PG_Ago2$parent_id == "Ago2"])),
linetype = "dashed", color = "black") +
        geom_vline(xintercept = max(end(PG_Ago2[PG_Ago2$parent_id == "Ago2"])), linetype
= "dashed", color = "black")

Ago2_1pOut <- Ago2originPiCzoomOut$trackAll$gtfNum1$trackPlus  +
        geom_vline(xintercept = min(start(PG_Ago2[PG_Ago2$parent_id == "Ago2"])),
linetype = "dashed", color = "black") +
        geom_vline(xintercept = max(end(PG_Ago2[PG_Ago2$parent_id == "Ago2"])), linetype
= "dashed", color = "black")
Ago2_1mOut <- Ago2originPiCzoomOut$trackAll$gtfNum1$trackMinus  +
        geom_vline(xintercept = min(start(PG_Ago2[PG_Ago2$parent_id == "Ago2"])),
linetype = "dashed", color = "black") +
        geom_vline(xintercept = max(end(PG_Ago2[PG_Ago2$parent_id == "Ago2"])), linetype
= "dashed", color = "black")
Ago2_2pOut <- Ago2originPiCzoomOut$trackAll$gtfNum2$trackPlus  +
        geom_vline(xintercept = min(start(PG_Ago2[PG_Ago2$parent_id == "Ago2"])),
linetype = "dashed", color = "black") +
        geom_vline(xintercept = max(end(PG_Ago2[PG_Ago2$parent_id == "Ago2"])), linetype
```

```
= "dashed", color = "black")
Ago2_2mOut <- Ago2originPiCzoomOut$trackAll$gtfNum2$trackMinus  +
        geom_vline(xintercept = min(start(PG_Ago2[PG_Ago2$parent_id == "Ago2"])),
linetype = "dashed", color = "black") +
        geom_vline(xintercept = max(end(PG_Ago2[PG_Ago2$parent_id == "Ago2"])), linetype
= "dashed", color = "black")
```

```
# Ago2 zoom in to Pseudogene region (+/- 50 bp)
message("Ago2-PG region: ", seqnames(PG_Ago2[PG_Ago2$parent_id == "Ago2"])[1], ":",
min(start(PG_Ago2[PG_Ago2$parent_id == "Ago2"])), "-", max(end(PG_Ago2[PG_Ago2$parent_id
== "Ago2"])))
chr <- seqnames(PG_Ago2[PG_Ago2$parent_id == "Ago2"])[1]
start <- min(start(PG_Ago2[PG_Ago2$parent_id == "Ago2"])) - 50
end <- max(end(PG_Ago2[PG_Ago2$parent_id == "Ago2"])) + 50
coord <- IRanges(start, end)

Ago2originPiCzoomIn <- allTracksPlotted(piRNAs_from_Bam = gr_mm10, chromosome = chr,
IRangesCoord=coord, gtfFiles=list(PG = PG_Ago2[PG_Ago2$parent_id == "Ago2"]),
tilesWidth=1, scaleWidthKB = 0.2)

# add dashed lines to plot to indicate pseudogene region
Ago2originPiCzoomInFull <- Ago2originPiCzoomIn$plotCoverageTrack  +
        geom_vline(xintercept = min(start(PG_Ago2[PG_Ago2$parent_id == "Ago2"])),
linetype = "dashed", color = "black") +
        geom_vline(xintercept = max(end(PG_Ago2[PG_Ago2$parent_id == "Ago2"])), linetype
= "dashed", color = "black")

Ago2_1pIn <- Ago2originPiCzoomIn$trackAll$gtfNum1$trackPlus  +
        geom_vline(xintercept = min(start(PG_Ago2[PG_Ago2$parent_id == "Ago2"])),
linetype = "dashed", color = "black") +
        geom_vline(xintercept = max(end(PG_Ago2[PG_Ago2$parent_id == "Ago2"])), linetype
= "dashed", color = "black")

Ago2_1mIn <- Ago2originPiCzoomIn$trackAll$gtfNum1$trackMinus  +
        geom_vline(xintercept = min(start(PG_Ago2[PG_Ago2$parent_id == "Ago2"])),
linetype = "dashed", color = "black") +
        geom_vline(xintercept = max(end(PG_Ago2[PG_Ago2$parent_id == "Ago2"])), linetype
= "dashed", color = "black")

Ago2originPiCFull <- Ago2_1pOut / Ago2_1mOut / Ago2_2pOut / Ago2_2mOut /
    Ago2originPiCzoomInFull / Ago2_1pIn / Ago2_1mIn

Ago2originPiCFull
```

```
Ago2-PG region: chr4:123838102-123839732

Normalizing to RPM
```



Pseudogene-Gene Sequence Identity

SPIN1 Pseudogene to Gene Comparison

```
# Pseudogene Spin1
message("Spin1-PG region: ", as.character(seqnames(PG_Spin1)[1]), ":",
min(start(PG_Spin1)), "-", max(end(PG_Spin1)))
chr <- seqnames(PG_Spin1)[1]
start <- min(start(PG_Spin1))
end <- max(end(PG_Spin1))

# Get sequence and save as fasta
genome <- BSgenome.Mmusculus.UCSC.mm10
Spin1_PG_seq <- DNAStringSet(getSeq(genome, chr, start, end))
names(Spin1_PG_seq) <- "Spin1_PG"
writeXStringSet(Spin1_PG_seq,
"../data/annotations/GeneVsPGcomparison/Spin1_Pseudogene_region.fasta")
Spin1_PG_seq
```

```
Spin1-PG region: chr9:67747306-67748766

DNAStringSet object of length 1:
    width seq                                                names
[1]  1461 TTTTTTTCAGATTTCTCAACAGT...CCACACTGGTCCAATGTTTTTCG Spin1_PG
```

```
# Gene Spin1
# get all alignments that are antisense to the Spin1 gene in the custom Protein Coding
Gene Transcriptome
Spin1_targeting_gr <- subsetByOverlaps(gr_alignments,
invertStrand(geneAnnotation_PCG_gene[geneAnnotation_PCG_gene$gene_id == "Spin1"]))
# then filter from those alignments all that origin from piC-as(Spin1) (in MILI
pachytene: rank 19)
Spin1_targetingFromPG_gr <- Spin1_targeting_gr[sub("_|_", "", sub("rank|_", "",
regmatches(Spin1_targeting_gr$qname, regexpr("rank(.*?)_", Spin1_targeting_gr$qname))))
== 19,]
```

```
# Retrieve DNA sequence from region where these alignments map to
PCG_fasta <-
"../data/annotations/customTranscriptomes/mm10_PCGtranscriptome_collapsed_prioritizedCDS
3UTR5UTR_allgenes.fasta"
PCG_seq <- readDNAStringSet(PCG_fasta)

target_region <- GRanges(
  seqnames = seqnames(Spin1_targetingFromPG_gr[1]),
  ranges = IRanges(start = min(start(Spin1_targetingFromPG_gr))-3, end =
max(end(Spin1_targetingFromPG_gr))+32)
)
message("Spin1-gene region: ",target_region)

Spin1_GENE_seq <- reverseComplement(PCG_seq[target_region])
names(Spin1_GENE_seq) <- "Spin1_GENE"
writeXStringSet(Spin1_GENE_seq,
"../data/annotations/GeneVsPGcomparison/Spin1_targeting_region_revcomp.fasta")
Spin1_GENE_seq
```

```
Spin1-gene region: chr13:1698750-1700210

DNAStringSet object of length 1:
    width seq                                               names
[1]  1461 TTTTACCCAGATTTCTCAACAGT...CCACACTGGTCCGATGTTTTCTG Spin1_GENE
```

```
# export sequences, run mulitple sequence alignments
Spin1_features <- sort(subsetByOverlaps(geneAnnotation_PCG[geneAnnotation_PCG$gene_id ==
"Spin1" & geneAnnotation_PCG$feature != "gene"], target_region))
Spin1_CDS3_seq <- reverseComplement(PCG_seq[Spin1_features[2]]) #1 would be the 3
nucleotides upstream from CDS3 but they don't match CDS2
Spin1_CDS4_seq <- reverseComplement(PCG_seq[Spin1_features[3]])
Spin1_CDS5_seq <- reverseComplement(PCG_seq[Spin1_features[4]])
Spin1_3UTR_seq <- reverseComplement(PCG_seq[Spin1_features[5]])

Spin1_all_seq <- c(Spin1_GENE_seq, Spin1_PG_seq, Spin1_3UTR_seq, Spin1_CDS5_seq,
Spin1_CDS4_seq, Spin1_CDS3_seq)
names(Spin1_all_seq) <- c("Spin1_GENE", "Spin1_PG", "Spin1_3UTR", "Spin1_CDS5",
"Spin1_CDS4", "Spin1_CDS3")
writeXStringSet(Spin1_all_seq,
"../data/annotations/GeneVsPGcomparison/Spin1_geneVsPG_all_regions.fasta")
```

```
# Feature annotations from UCSC Genome Browser
Spin1_GENE_coord <- GRanges(
  seqnames = rep("Spin1_GENE", 4),
  ranges = IRanges(
    start = c(1, 771, 971, 1205),
    end = c(770, 970, 1204, 1458)
  ),
  feature_type = c("3' UTR", "Exon 5", "Exon 4", "Exon 3"),
  organism = rep("Spin1_GENE", 4)
)

Spin1_PG_coord <- GenomicRanges::shift(PG_Spin1, (-min(start(PG_Spin1))+1))
Spin1_PG_coord <- renameSeqlevels(Spin1_PG_coord, c("chr9" = "Spin1_PG", "chr13" =
"Spin1_GENE"))
Spin1_PG_coord$organism <- rep("Spin1_PG", length(Spin1_PG_coord))
Spin1_GENE_PG_coord <- c(Spin1_GENE_coord, Spin1_PG_coord)
```

```
save(Spin1_GENE_PG_coord, file =
"../data/annotations/GeneVsPGcomparison/Spin1_GENE_PG_coord.RData")
```

BASH

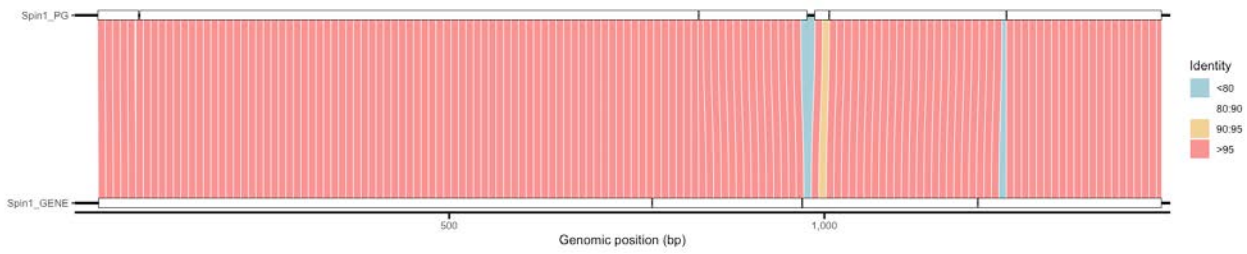
```
minimap2 -x asm10 -c -eqx -secondary=no \
    ../data/annotations/GeneVsPGcomparison/Spin1_targeting_region_revcomp.fasta \
    ../data/annotations/GeneVsPGcomparison/Spin1_Pseudogene_region.fasta \
    > ../data/annotations/GeneVsPGcomparison/Spin1vsSpin1PG.paf
```

R

In case you switched your kernel, rerun the *Prep*-section of this notebook first

```
options(repr.plot.width=15, repr.plot.height=3)
# read paf file (output of minimap2), using SVbyEye package
paf.table <- readPaf(
    paf.file = "../data/annotations/GeneVsPGcomparison/Spin1vsSpin1PG.paf",
    include.paf.tags = TRUE#, restrict.paf.tags = "cg"
)

# Read feature annotations (PG: prediction)
annot.gr <-
get(load("../data/annotations/GeneVsPGcomparison/Spin1_GENE_PG_coord.RData"))

# Plot identity for Spin1 annotation
plt_fullPiC <- plotAVA(
    paf.table = paf.table,
    color.by = "identity",
    binsize = 10, perc.identity.breaks = c(80, 90, 95)
)

# Add feature annotations to plot
plt_fullPiCwAnn <- addAnnotation(
    ggplot.obj = plt_fullPiC,
    annot.gr = annot.gr,
    coordinate.space = "self",
    shape = "rectangle",
    y.label.id = "organism",
    annotation.level = 0
)

plt_fullPiCwAnn
```

```
[readPaf] Loading PAF file: ../data/annotations/GeneVsPGcomparison/Spin1vsSpin1P
G.paf
 ... 0s

[pafToBins] Binning PAF alignments, binsize=10bp
 ... 0.08s
```



Ago2 Pseudogene to Gene Comparison

```
# Get Ago2 pseudogene sequence and annotation prediction
PG_Ago2 <- pseudogenesRGOvrlp[pseudogenesRGOvrlp$parent_id == "Ago2"]
PG_Ago2_seq <- DNAStringSet(getSeq(BSgenome.Mmusculus.UCSC.mm10, GRanges(seqnames =
seqnames(PG_Ago2[1]), ranges = IRanges(start = min(start(PG_Ago2))-tailing_len, end =
max(end(PG_Ago2))+tailing_len))))
names(PG_Ago2_seq) <- "Ago2_PG"
PG_Ago2_seq
Biostrings::writeXStringSet(PG_Ago2_seq,
"../data/annotations/GeneVsPGcomparison/Ago2_Pseudogene_region.fasta", format = "fasta")

Ago2_PG_NULL <- PG_Ago2
Ago2_PG_NULL$organism <- "Ago2_PG"
Ago2_PG_NULL <- renameSeqlevels(Ago2_PG_NULL, c("chr4" = "Ago2_PG"))
start(Ago2_PG_NULL) <- start(Ago2_PG_NULL) - min(start(PG_Ago2)) + 1
end(Ago2_PG_NULL)   <- end(Ago2_PG_NULL)   - min(start(PG_Ago2)) + 1
```

```
Error: object 'pseudogenesRGOvrlp' not found
Traceback:
```

```
# get Ago2 gene
gene_Ago2 <- sort(genes[genes$gene_id == "Ago2", ], decreasing = TRUE)
chr <- as.character(seqnames(gene_Ago2[1]))
# +1 indexing for exons, bc 1st exon is annotated as feature 5UTR and CDS separately
ex <- function(i) gene_Ago2[i + 1]  # your object uses +1 indexing for exons

# CDS exons 11–18
get_exon_seq <- function(i) {
  s <- DNAStringSet(
    reverseComplement(
      getSeq(BSgenome.Mmusculus.UCSC.mm10,
             GRanges(seqnames = chr,
                     ranges = IRanges(start = start(ex(i)),
                                      end   = end(ex(i))))))
    )
  )
  names(s) <- paste0("exon_", i)
  s
}
exon_seqs <- do.call(c, lapply(11:18, get_exon_seq))

# Intron between exon18 and exon19
```

```
intron18_seq <- DNAStringSet(
  reverseComplement(
    getSeq(BSgenome.Mmusculus.UCSC.mm10,
           GRanges(seqnames = chr,
                   ranges = IRanges(start = end(ex(19)) + 1,
                                    end   = start(ex(18)) - 1)))
  )
)
names(intron18_seq) <- "intron_18"

# exon19 CDS + beginning of 3'UTR, until 73106583
ex19_start <- 73106583 # coordinate through BLAT of Ago2 PG to gene in UCSC Genome
Browser
exon19part_seq <- DNAStringSet(
  reverseComplement(
    getSeq(BSgenome.Mmusculus.UCSC.mm10,
           GRanges(seqnames = chr,
                   ranges = IRanges(start = ex19_start,
                                    end   = end(ex(19)))))
  )
)
names(exon19part_seq) <- "exon_19_part"

# Combine in transcript order
gene_Ago2_part_seq <- c(exon_seqs, intron18_seq, exon19part_seq)

# Build the GRanges using the sequence widths
lens <- width(gene_Ago2_part_seq)
offsets <- cumsum(c(0, lens[-length(lens)]))
gene_Ago2NULL <- GRanges(
    seqnames = "Ago2_genePartial",
    ranges   = IRanges(start = offsets + 1, width = lens),
    strand   = "+",
    gene_name = names(gene_Ago2_part_seq)
)
mcols(gene_Ago2NULL)$organism <- "Ago2_genePartial"

# Put sequence together and save as fasta
gene_Ago2_partCom_seq <- DNAStringSet(paste0(as.character(gene_Ago2_part_seq), collapse
= ""))
names(gene_Ago2_partCom_seq) <- "Ago2_genePartial"
gene_Ago2_partCom_seq
Biostrings::writeXStringSet(gene_Ago2_partCom_seq,
"../data/annotations/GeneVsPGcomparison/Ago2_genePartial.fasta", format = "fasta")

# save GRanges object
gene_Ago2NULL <- gene_Ago2NULL[!gene_Ago2NULL$gene_name == "intron_18"]
```

```
DNAStringSet object of length 1:
    width seq                                               names
[1]  1784 AACAAAGCAATTGCCACCCCTGT...GAACTCTCAGGGCTTTAAAACAC Ago2_genePartial
```

```
gr_exonsAgo2 <- c(Ago2_PG_NULL, gene_Ago2NULL)
save(gr_exonsAgo2, file =
"../data/annotations/GeneVsPGcomparison/Ago2_gene_PG_corrd.RData")
```

```
Warning message in .merge_two_Seqinfo_objects(x, y):
"The 2 combined objects have no sequence levels in common. (Use
  suppressWarnings() to suppress this warning.)"
```

BASH

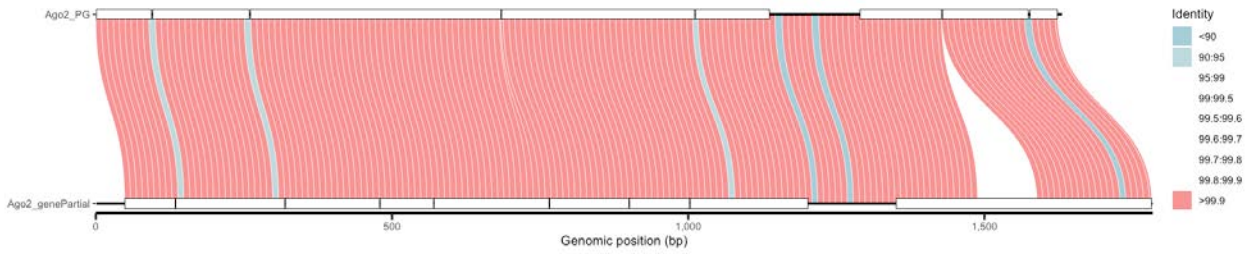
```
minimap2 -x asm10 -c -eqx -secondary=no \
    ../data/annotations/GeneVsPGcomparison/Ago2_genePartial.fasta \
    ../data/annotations/GeneVsPGcomparison/Ago2_Pseudogene_region.fasta \
    > ../data/annotations/GeneVsPGcomparison/Ago2vsAgo2PG.paf
```

R

In case you switched your kernel, rerun the *Prep*-section of this notebook first

```
options(repr.plot.width=15, repr.plot.height=3)
# read paf file (output of minimap2), using SVbyEye package
paf.table <- readPaf(
    paf.file = "../data/annotations/GeneVsPGcomparison/Ago2vsAgo2PG.paf",
    include.paf.tags = TRUE#, restrict.paf.tags = "cg"
)
# Read exon annotations (PG: prediction)
annot.gr <- get(load("../data/annotations/GeneVsPGcomparison/Ago2_gene_PG_corrd.RData"))

# Plot identity for Ago2 annotation
plt_fullPiC <- plotAVA(
    paf.table = paf.table,
    color.by = "identity",
    binsize = 10
)

plt_fullPiCwAnn <- addAnnotation(
    ggplot.obj = plt_fullPiC,
    annot.gr = annot.gr,
    coordinate.space = "self",
    shape = "rectangle",
    y.label.id = "organism",
    annotation.level = 0
)
plt_fullPiCwAnn
```

```
[readPaf] Loading PAF file: EvoComparison/AGO2PG/Ago2vsAgo2PG.paf
 ... 0s

[pafToBins] Binning PAF alignments, binsize=10bp
 ... 0.12s
```



ECDF (Fig 1d, Extended Data Fig. 1c)

```
options(repr.plot.width=9, repr.plot.height=6)

# Generic func to get featureCounts and DESeq2 results
calcECDF <- function(bam_files, ann_dir, sample_name) {
```

```
    # Run featureCounts for gene counts
    counts <- featureCounts(
        files = bam_files,
        annot.ext = ann_dir,
        isGTFAnnotationFile = TRUE,
        GTF.featureType = "exon",
        GTF.attrType = "gene_id",
        useMetaFeatures = TRUE,
        nthreads = 8,
        isPairedEnd = FALSE
    )

    # shorter sample names (drop trailing mapping suffixes)
    sample_ids <- sub("\\.fastq.*$", "", basename(colnames(counts$counts)))   # e.g.
MIWI_HET1_SRR610421
    colnames(counts$counts) <- sample_ids

    # derive condition (HET/KO) and replicate from sample names
    condition <- factor(sub(".*_(HET|KO)[0-9]+_.*", "\\1", sample_ids))
    message("Condition: ", condition)
    condition <- relevel(condition, ref = "HET")         # HET as control/reference
    replicate <- as.integer(sub(".*_(?:HET|KO)([0-9]+)_.*", "\\1", sample_ids))


    # sample info
    coldata <- data.frame(
    sample = sample_ids,
    condition = condition,
    replicate = replicate,
    row.names = sample_ids,
    check.names = FALSE
    )

    # DESeq2 pipeline
    dds <- DESeqDataSetFromMatrix(
    countData = counts$counts,
    colData = coldata,
    design = ~ condition
    )
    dds <- DESeq(dds)

    # pick the correct coefficient name
    coef_name <- grep("^condition_.*KO.*_vs_.*HET$", resultsNames(dds), value = TRUE)
    stopifnot(length(coef_name) == 1)

    # LFC shrinkage and convert to data frame
    res <- lfcShrink(dds, coef = coef_name, type = "apeglm")
    res_df <- as.data.frame(res)
    gene_ids <- rownames(res_df)
    lfc <- data.frame(
        gene   = gene_ids,
        log2FC = res_df$log2FoldChange,
        row.names = NULL,
        check.names = FALSE
    )

    return(list(dds, lfc, res))
}
```

```
top_targeted <-
targetingByPiRNAsSbO_sorted_total_tra[1:genes_targetingThreshold,]$geneName
length(top_targeted)
# Filter rows where totalPiRNAcount is greater than 0 and exclude rows 1 to 88
(genes_targetingThreshold)
mid_targeted <- targetingByPiRNAsSbO_sorted_total_tra[
  (targetingByPiRNAsSbO_sorted_total_tra$totalPiRNAcount > 0) &
  !(seq_len(nrow(targetingByPiRNAsSbO_sorted_total_tra)) %in%
1:genes_targetingThreshold),
]$geneName
length(mid_targeted)
not_targeted <-
targetingByPiRNAsSbO_sorted_total_tra[targetingByPiRNAsSbO_sorted_total_tra$totalPiRNAco
unt == 0,]$geneName
length(not_targeted)
```

```
88
```

```
9622
```

```
11799
```

```
# Sample: Late SPC, get bam files
bam_files_lateSPC <- c(
  "MIWI_0/lateSPC/MIWI_HET1_SRR610673.fastq.gz.mapped.Aligned.sortedByCoord.out.bam",
  "MIWI_0/lateSPC/MIWI_HET2_SRR610674.fastq.gz.mapped.Aligned.sortedByCoord.out.bam",
  "MIWI_0/lateSPC/MIWI_HET3_SRR610675.fastq.gz.mapped.Aligned.sortedByCoord.out.bam",
  "MIWI_0/lateSPC/MIWI_KO1_SRR610676.fastq.gz.mapped.Aligned.sortedByCoord.out.bam",
  "MIWI_0/lateSPC/MIWI_KO2_SRR610677.fastq.gz.mapped.Aligned.sortedByCoord.out.bam",
  "MIWI_0/lateSPC/MIWI_KO3_SRR610678.fastq.gz.mapped.Aligned.sortedByCoord.out.bam"
)
ann_dir <-
"../../../OneDrive/General/mmu_referenceGenome/Mm10_refSeq3_copies_annotated3.sorted.gtf
"

# run function above to get featureCounts and DESeq2 results and LFC shrinkage
dds_lfc_lateSPC <- calcECDF(bam_files_lateSPC, ann_dir, "lateSPC")
dds_lateSPC <- dds_lfc_lateSPC[1][[1]]
lfc_lateSPC <- dds_lfc_lateSPC[2][[1]]
res_lateSPC <- dds_lfc_lateSPC[3][[1]]
```

```
        ==========     _____ _    _ ____  _____  ______          _____
        =====         / ____| |  | |  _ \|  __ \|  ____|   /\   |  __ \
          =====      | (___ | |  | | |_) | |__) | |__     /  \  | |  | |
            ====      \___ \| |  | |  _ <|  _  /|  __|   / /\ \ | |  | |
              ====    ____) | |__| | |_) | | \ \| |____ / ____ \| |__| |
        ==========   |_____/ \____/|____/|_|  \_\______/_/    \_\_____/
       Rsubread 2.20.0

//========================== featureCounts setting ===========================\\
||                                                                            ||
||             Input files : 6 BAM files                                      ||
||                                                                            ||
||                           MIWI_HET1_SRR610673.fastq.gz.mapped.Aligned. ... ||
||                           MIWI_HET2_SRR610674.fastq.gz.mapped.Aligned. ... ||
||                           MIWI_HET3_SRR610675.fastq.gz.mapped.Aligned. ... ||
||                           MIWI_KO1_SRR610676.fastq.gz.mapped.Aligned.s ... ||
||                           MIWI_KO2_SRR610677.fastq.gz.mapped.Aligned.s ... ||
||                           MIWI_KO3_SRR610678.fastq.gz.mapped.Aligned.s ... ||
||                                                                            ||
```

```
||              Paired-end : no                                                ||
||        Count read pairs : no                                                ||
||              Annotation : Mm10_refSeq3_copies_annotated3.sorted.gtf (GTF)   ||
||      Dir for temp files : .                                                 ||
||                 Threads : 8                                                 ||
||                   Level : meta-feature level                                ||
||      Multimapping reads : counted                                           ||
|| Multi-overlapping reads : not counted                                       ||
||   Min overlapping bases : 1                                                 ||
||                                                                             ||
\\=============================================================================//

//================================= Running ===================================\\
||                                                                             ||
|| Load annotation file Mm10_refSeq3_copies_annotated3.sorted.gtf ...          ||
||    Features : 1366783                                                       ||
||    Meta-features : 49605                                                    ||
||    Chromosomes/contigs : 21                                                 ||
||                                                                             ||
|| Process BAM file MIWI_HET1_SRR610673.fastq.gz.mapped.Aligned.sortedByC ...  ||
||    Single-end reads are included.                                           ||
||    Total alignments : 134379273                                             ||
||    Successfully assigned alignments : 116179920 (86.5%)                    ||
||    Running time : 0.50 minutes                                              ||
||                                                                             ||
|| Process BAM file MIWI_HET2_SRR610674.fastq.gz.mapped.Aligned.sortedByC ...  ||
||    Single-end reads are included.                                           ||
||    Total alignments : 111409874                                             ||
||    Successfully assigned alignments : 96245345 (86.4%)                     ||
||    Running time : 0.38 minutes                                              ||
||                                                                             ||
|| Process BAM file MIWI_HET3_SRR610675.fastq.gz.mapped.Aligned.sortedByC ...  ||
||    Single-end reads are included.                                           ||
||    Total alignments : 152438457                                             ||
||    Successfully assigned alignments : 126889984 (83.2%)                    ||
||    Running time : 0.51 minutes                                              ||
||                                                                             ||
|| Process BAM file MIWI_KO1_SRR610676.fastq.gz.mapped.Aligned.sortedByCo ...  ||
||    Single-end reads are included.                                           ||
||    Total alignments : 110503411                                             ||
||    Successfully assigned alignments : 92847870 (84.0%)                     ||
||    Running time : 0.37 minutes                                              ||
||                                                                             ||
|| Process BAM file MIWI_KO2_SRR610677.fastq.gz.mapped.Aligned.sortedByCo ...  ||
||    Single-end reads are included.                                           ||
||    Total alignments : 151008781                                             ||
||    Successfully assigned alignments : 127852629 (84.7%)                    ||
||    Running time : 0.50 minutes                                              ||
||                                                                             ||
|| Process BAM file MIWI_KO3_SRR610678.fastq.gz.mapped.Aligned.sortedByCo ...  ||
||    Single-end reads are included.                                           ||
||    Total alignments : 106079594                                             ||
||    Successfully assigned alignments : 88100959 (83.1%)                     ||
||    Running time : 0.35 minutes                                              ||
||                                                                             ||
|| Write the final count table.                                                ||
|| Write the read assignment summary.                                          ||
||                                                                             ||
\\=============================================================================//

Condition: HETHETHETKOKOKO

estimating size factors

estimating dispersions
```

```
        gene-wise dispersion estimates

        mean-dispersion relationship

        final dispersion estimates

        fitting model and testing

        using 'apeglm' for LFC shrinkage. If used in published research, please cite:
            Zhu, A., Ibrahim, J.G., Love, M.I. (2018) Heavy-tailed prior distributions f
        or
            sequence count data: removing the noise and preserving large differences.
            Bioinformatics. https://doi.org/10.1093/bioinformatics/bty895
```

```r
# Build a long table of log2FC for each group (drop NAs and genes not in lfc)
sets <- list(
  `Top targeted` = unique(top_targeted),
  `Mid targeted` = unique(mid_targeted),
  `Not targeted` = unique(not_targeted)
)
ecdf_df <- bind_rows(lapply(names(sets), function(grp) {
  tibble(
    gene = sets[[grp]]
  ) %>%
    inner_join(as_tibble(lfc_lateSPC), by = "gene") %>%
    mutate(group = grp)
})) %>%
  filter(!is.na(log2FC)) %>%
  mutate(group = factor(group, levels = c("Top targeted","Mid targeted","Not
targeted")))

#Number of genes per group
table(ecdf_df$group)

# ECDF plot
# get log2FC (x-axis) and ecdf (y-axis) values for Spin1 and Ago2
ecdf_df_tT <- subset(ecdf_df, group == "Top targeted")
ecdf_function <- ecdf(ecdf_df_tT$log2FC)
highlight_df <- data.frame(
  gene = c("Ago2", "Spin1"),
  log2FC = c(
    ecdf_df_tT[ecdf_df_tT$gene == "Ago2",]$log2FC,
    ecdf_df_tT[ecdf_df_tT$gene == "Spin1",]$log2FC),
  ecdf_y = ecdf_function(c(ecdf_df_tT[ecdf_df_tT$gene == "Ago2",]$log2FC,
    ecdf_df_tT[ecdf_df_tT$gene == "Spin1",]$log2FC)),
  group = "Top targeted"
)
highlight_df

# plot ECDF
ECDF_lateSPC <- ggplot(ecdf_df, aes(x = log2FC, colour = group)) +
  stat_ecdf(size = 1) +
  scale_colour_manual(
    values = c(
      "Top targeted" = "#9865aa",
      "Mid targeted" = "#7771b3",
      "Not targeted" = "#231f20"
    )
  ) +
```

```
  geom_point(data = highlight_df, aes(x = log2FC, y = ecdf_y), fill = "#9764aa", color =
"black", stroke = 0.2, size = 1) +
  geom_text(data = highlight_df, aes(x = log2FC, y = ecdf_y, label = gene),
            color = "#9764aa", vjust = 1.8, hjust = -0.1, size = 3) +
  labs(x = "log2 fold-change (MIWI KO vs HET)", y = "ECDF", colour = "Group") +
  coord_cartesian(xlim = c(-0.3, 1.8)) +
  theme_classic() +
  ggtitle("Late SPC") +
  theme(axis.title.x = element_text(size = 7),
        axis.title.y = element_text(size = 7),
        axis.text.x = element_text(size = 7),
        axis.text.y = element_text(size = 7),
        legend.text = element_text(size = 7),
        legend.title = element_text(size = 7))

ECDF_lateSPC


# Pairwise Kolmogorov-Smirnov Test
groups <- split(ecdf_df$log2FC, ecdf_df$group)
combn(names(groups), 2, FUN = function(p) {
  data.frame(
    pair = paste(p, collapse = " vs "),
    ks_p = ks.test(groups[[p[1]]], groups[[p[2]]])$p.value
  )
}, simplify = FALSE) |> dplyr::bind_rows()
```

```
Top targeted Mid targeted Not targeted
          86          9288          9497
```

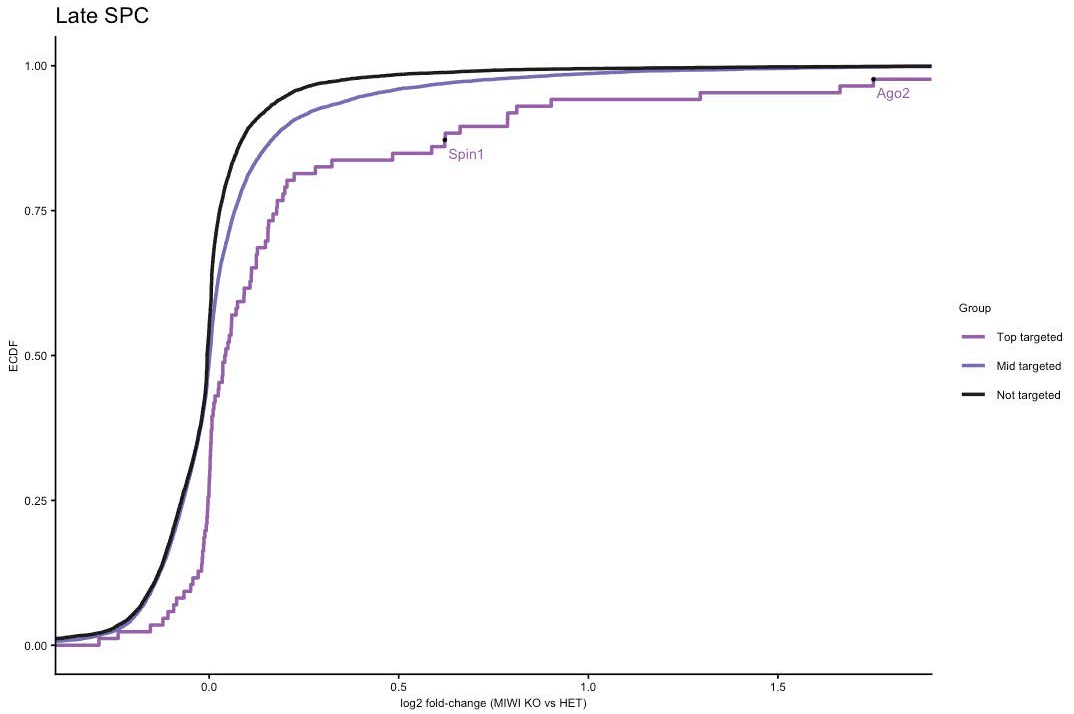
A data.frame: 2 x 4

gene	log2FC	ecdf_y	group
<chr>	<dbl>	<dbl>	<chr>
Ago2	1.7517264	0.9767442	Top targeted
Spin1	0.6214549	0.8720930	Top targeted

```
Warning message in ks.test.default(groups[[p[1]]], groups[[p[2]]]):
"p-value will be approximate in the presence of ties"
Warning message in ks.test.default(groups[[p[1]]], groups[[p[2]]]):
"p-value will be approximate in the presence of ties"
Warning message in ks.test.default(groups[[p[1]]], groups[[p[2]]]):
"p-value will be approximate in the presence of ties"
```

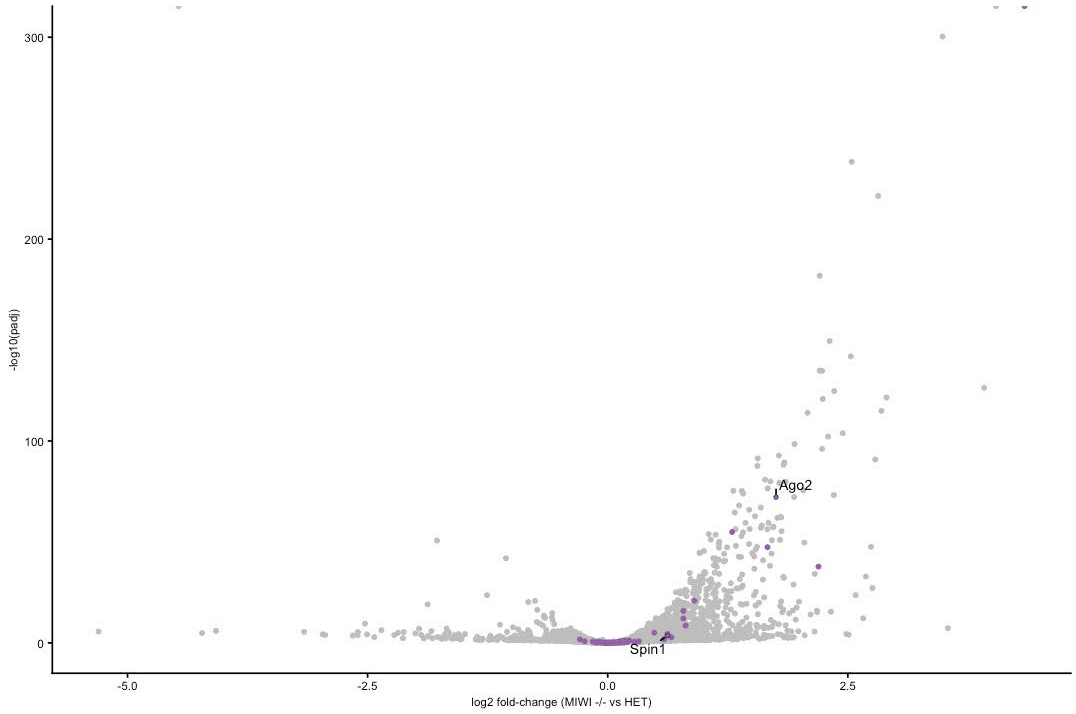
A data.frame: 3 x 2

pair	ks_p
<chr>	<dbl>
Top targeted vs Mid targeted	2.736105e-05
Top targeted vs Not targeted	9.061781e-08
Mid targeted vs Not targeted	3.072089e-48




```
# Combine calcECDF-outputs to get a combined dataframe wiht gene name, log2FC, padj
coef_name <- grep("^condition_.*KO.*_vs_.*HET$", resultsNames(dds_lateSPC), value =
TRUE)
res_full <- results(dds_lateSPC, name = coef_name)

tbl <- as.data.frame(res_lateSPC) %>%
  transmute(gene = rownames(res_lateSPC),
            log2FC = log2FoldChange) %>%
  left_join(
    as.data.frame(res_full) %>%
      transmute(gene = rownames(res_full),
                padj = padj),
    by = "gene"
  ) %>%
  filter(!is.na(log2FC), !is.na(padj))

# create grouping in tbl: top targeted vs all others (mid- and non-targets)
top_targ <- unique(top_targeted)
tbl <- tbl %>%
  mutate(group = ifelse(gene %in% top_targ, "Top targeted", "Other"),
         is_callout = gene %in% c("Spin1","Ago2"))

#plot volcano
p_volcano <- ggplot(tbl, aes(x = log2FC, y = -log10(padj))) +
  # mid and not targeted genes (Other) in grey
  geom_point(data = subset(tbl, group == "Other"),
             color = "grey75", size = 1) +
  # top targeted in purple
  geom_point(data = subset(tbl, group == "Top targeted" & !is_callout),
             color = "#9764aa", size = 1) +
  # highlight Spin1 and Ago2
  geom_point(data = subset(tbl, is_callout),
             aes(fill = group),
             shape = 21, color = "black", stroke = 0.2, size = 1.2, show.legend = FALSE)
+
  scale_fill_manual(values = c("Top targeted" = "#9764aa", "Other" = "grey75")) +
```

```
  geom_text_repel(data = subset(tbl, is_callout),
                  aes(label = gene),
                  size = 3, box.padding = 0.25, point.padding = 0.2,
                  min.segment.length = 0, max.overlaps = Inf) +
  labs(x = "log2 fold-change (MIWI -/- vs HET)",
       y = "-log10(padj)") +
  theme_classic() +
  theme(axis.title.x = element_text(size = 7),
        axis.title.y = element_text(size = 7),
        axis.text.x = element_text(size = 7),
        axis.text.y = element_text(size = 7),
        legend.text = element_text(size = 7),
        legend.title = element_text(size = 7))

p_volcano
```



piC-as(Spin1) and piC-as(Ago2) RNA seq Analysis

Code for Manuscript Figure 2c,d and Extended Data Figure 3f

Introduction to files

R/4.4.3 and Bash used

Samples used for paired-end 50bp RNA seq:

- 161602 piC-as(Spin1) KO
- 161603 piC-as(Spin1) KO
- 161604 piC-as(Spin1) KO
- 1069 piC-as(Ago2) KO
- 1070 piC-as(Ago2) KO
- 1071 piC-as(Ago2) KO
- 161212 WT
- 161213 WT
- 161922 WT

Mapping the fastq files_bash

```
#!/bin/bash
#STAR mapping and indexing of paired-end sequencing data

input_file1=$1
input_file2=$2

basename=$(basename "$input_file1")
output_prefix="${OUTDIR}/$(basename "$input_file1").mapped."

module load STAR/2.7.10b
module load samtools/1.17

mkdir -p "$OUTDIR"

STAR \
  --readFilesIn "$input_file1" "$input_file2" \
  --readFilesCommand gunzip -c \
  --genomeDir "$GENOME_DIR" \
  --runThreadN 12 \
  --genomeLoad LoadAndRemove \
  --limitBAMsortRAM 20000000000 \
  --outFileNamePrefix "$output_prefix" \
  --outSAMtype BAM SortedByCoordinate \
  --outReadsUnmapped Fastx \
  --outFilterMultimapNmax 100 \
  --outFilterMultimapScoreRange 0 \
  --outFilterMismatchNoverLmax 0.05 \
  --sjdbScore 2
```

```
cd "$OUTDIR"
samtools index -@ 24 -M -b *.bam

#SLURM batch submission
#Each job processes a pair of FASTQ files (_R1_ and _R2_).

: '
for forward_file in ${WORKDIR}/*_R1_001.fastq.gz; do
  sample_name=$(basename "$forward_file" "_R1_001.fastq.gz")
  reverse_file="${WORKDIR}/${sample_name}_R2_001.fastq.gz"
  echo "Submitting job for sample: $sample_name"

  sbatch --mem=30g --cpus-per-task=16 --time=5:00:00 \
    ./mapping_mm10_pipeline.sh "$forward_file" "$reverse_file"
done
'
```

FeatureCounts to count gene expression_bash

```
#!/bin/bash
#FeatureCounts for gene expression

input_file1=$1

basename=$(basename "$input_file1")
output_prefix1="${OUTDIR}/$(basename "$input_file1")_featureCounts"

mkdir -p "$OUTDIR"

module load subread

featureCounts \
  -p \
  --countReadPairs \
  -s 2 \
  -a "$ANNOTATION" \
  -o "${output_prefix1}.txt" \
  $input_file1;

#SLURM batch submission
#Each BAM file will be submitted as a separate SLURM job

: '
for file in ${WORKDIR}/*.bam; do
  sample_name=$(basename "$file")
  echo "Submitting featureCounts job for: $sample_name"
  sbatch --mem=10g --cpus-per-task=8 \
    ./featureCounts_refseq.sh "$file"
done
'
```

FeatureCounts to count TE expression_bash

```
#!/bin/bash
#FeatureCounts for TE expression

input_file1=$1
```

```
basename=$(basename "$input_file1")
output_prefix1="${OUTDIR}/$(basename "$input_file1")_featureCounts_TE"

mkdir -p "$OUTDIR"

module load subread

featureCounts \
  -p \
  --countReadPairs \
  -s 2 \
  -g gene_id \
  -M \
  -O \
  --fraction \
  -a "$ANNOTATION_TE" \
  -o "${output_prefix1}.txt" \
  $input_file1;

#SLURM batch submission
#Each BAM file will be submitted as a separate SLURM job

: '
for file in ${WORKDIR}/*.bam; do
  sample_name=$(basename "$file")
  echo "Submitting featureCounts job for: $sample_name"
  sbatch --mem=10g --cpus-per-task=8 \
    ./featureCounts_TE.sh "$file"
done
'
```

Analyze gene expression in piC-as(Ago2) vs WT mice_Fig. 2c

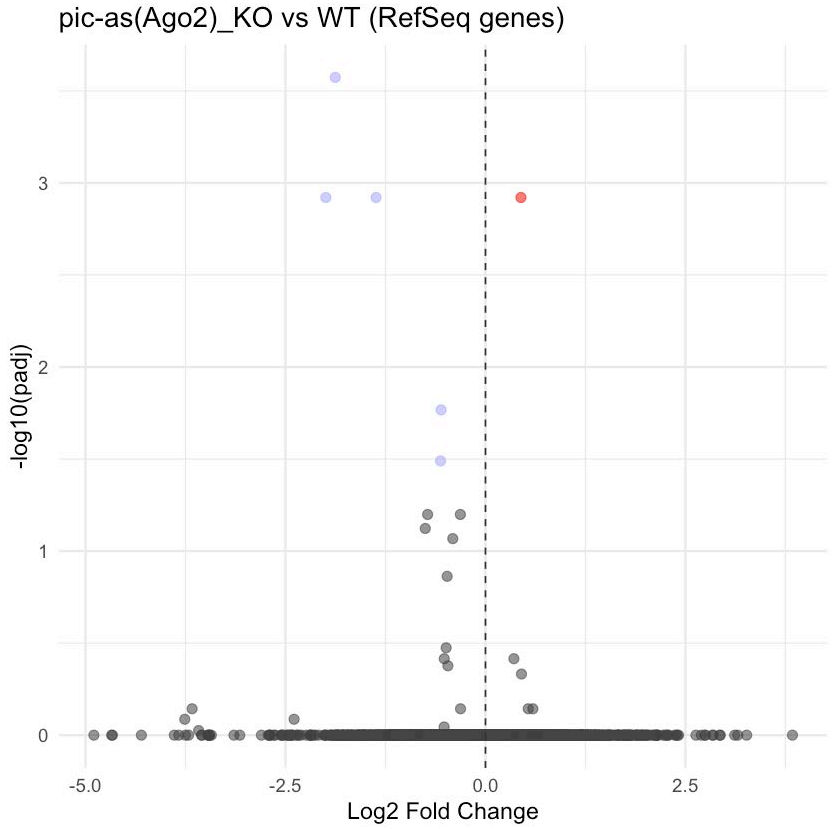
```
#   Suppress warnings globally   #

options(warn = -1)
suppressPackageStartupMessages({
  library(DESeq2)
  library(GenomicFeatures)
  library(ggplot2)
  library(writexl)
})
```

```
################################################################################
# Title: Differential Expression Analysis of RefSeq Genes in pic-as(Ago2) Knockout
Testes
# Description: This script performs DESeq2 analysis of RefSeq-annotated RNA-seq
#              data from WT and pic-as(Ago2) knockout mouse testes and generates a
#              volcano plot of differential expression results.
################################################################################


#    Input files    #


base_dir <- "/Users/loubalovaz2/Documents/Lab/Pachytene piRNA
project/Library_prep/240220_Spin1, Ago2, WT - RNAseq_whole
```

```
testis/Long_RNA_libraries/Proper_50_PE_seq/featureCountes_refseq"
output_dir <- "/Users/loubalovaz2/Documents/Lab/Writing/Pachytene_clusters
paper/Code_results"

count_files <- c(
  file.path(base_dir,"161212_Zuzana-
7repeat_S2_L008_R1_001.fastq.gz.mapped.Aligned.sortedByCoord.out.bam_featureCounts.txt")
,

file.path(base_dir,"161213_s08_S6_R1_001.fastq.gz.mapped.Aligned.sortedByCoord.out.bam_f
eatureCounts.txt"),

file.path(base_dir,"161922_s09_S7_R1_001.fastq.gz.mapped.Aligned.sortedByCoord.out.bam_f
eatureCounts.txt"),

file.path(base_dir,"1069_s04_S3_R1_001.fastq.gz.mapped.Aligned.sortedByCoord.out.bam_fea
tureCounts.txt"),

file.path(base_dir,"1070_s05_S4_R1_001.fastq.gz.mapped.Aligned.sortedByCoord.out.bam_fea
tureCounts.txt"),

file.path(base_dir,"1071_s06_S5_R1_001.fastq.gz.mapped.Aligned.sortedByCoord.out.bam_fea
tureCounts.txt")
)


#   Load and format counts  #


count_list <- lapply(count_files, function(file) {
  read.table(file, header = TRUE, row.names = 1)
})

# Combine into one matrix and select count columns
count_matrix_all <- do.call(cbind, count_list)
count_matrix <- count_matrix_all[, c(6, 12, 18, 24, 30, 36)]

# Rename columns
colnames(count_matrix) <- c("WT_1", "WT_2", "WT_3", "Ago2_1", "Ago2_2", "Ago2_3")


#   Define sample metadata  #


sample_metadata <- data.frame(
  row.names = colnames(count_matrix),
  Genotype = c("WT", "WT", "WT", "Ago2_KO", "Ago2_KO", "Ago2_KO")
)

# Verify that sample names match between counts and metadata
stopifnot(all(colnames(count_matrix) %in% rownames(sample_metadata)))
stopifnot(all(colnames(count_matrix) == rownames(sample_metadata)))


#      DESeq2 analysis      #


dds <- DESeqDataSetFromMatrix(
  countData = count_matrix,
  colData = sample_metadata,
```

```
  design = ~ Genotype
)

# Prefilter low-count genes
dds <- dds[rowSums(counts(dds)) >= 10, ]

# Set reference level
dds$Genotype <- relevel(dds$Genotype, ref = "WT")

# Run DESeq2
dds <- DESeq(dds)
res <- results(dds)
res_df <- as.data.frame(res)


#   Add gene length filter as the RNA isolation was prepared wth >200bp cut off   #
txdb <-
makeTxDbFromGFF("/Users/loubalovaz2/Documents/Genomes/Mouse/Annotation/mm10_refGene.gtf"
, format = "gtf")
gene_gr <- genes(txdb)
gene_length <- data.frame(gene_id = names(gene_gr), length = width(gene_gr))

# Merge DE results with gene lengths
res_df$gene_id <- rownames(res_df)
res_df <- merge(res_df, gene_length, by = "gene_id")

# Filter: keep genes >200 bp as the RNA isolation method used columns with 200bp cut off
keep_counts <- rowSums(counts(dds) > 10) == ncol(dds)
res_df <- subset(res_df, keep_counts & length > 200)

# Save results to Excel
write_xlsx(res_df, file.path(output_dir,
"Ago2_Refseq_DESeq2_results_10reads_200nt.xlsx"))


#       Volcano plot        #


res_df$color <- "#484545"
res_df$color[res_df$padj < 0.05 & res_df$log2FoldChange > 0] <- "#f80000"
res_df$color[res_df$padj < 0.05 & res_df$log2FoldChange < 0] <- "#aeaefc"

volcano_plot_Ago2 <- ggplot(res_df, aes(x = log2FoldChange, y = -log10(padj), color =
color)) +
  geom_point(alpha = 0.6, size = 2.5) +
  geom_vline(xintercept = 0, linetype = "dashed", color = "#343333") +
  scale_color_identity() +
  labs(
    x = "Log2 Fold Change",
    y = "-log10(padj)",
    title = "pic-as(Ago2)_KO vs WT (RefSeq genes)"
  ) +
  theme_minimal(base_size = 14)

ggsave(file.path(output_dir,"Ago2_refseq_volcano_10reads_200nt.svg"), plot =
volcano_plot_Ago2, bg = "white", width = 6, height = 5)
ggsave(file.path(output_dir,"Ago2_refseq_volcano_10reads_200nt.png"), plot =
volcano_plot_Ago2, bg = "white", width = 6, height = 5)
```

```
print(volcano_plot_Ago2)
```



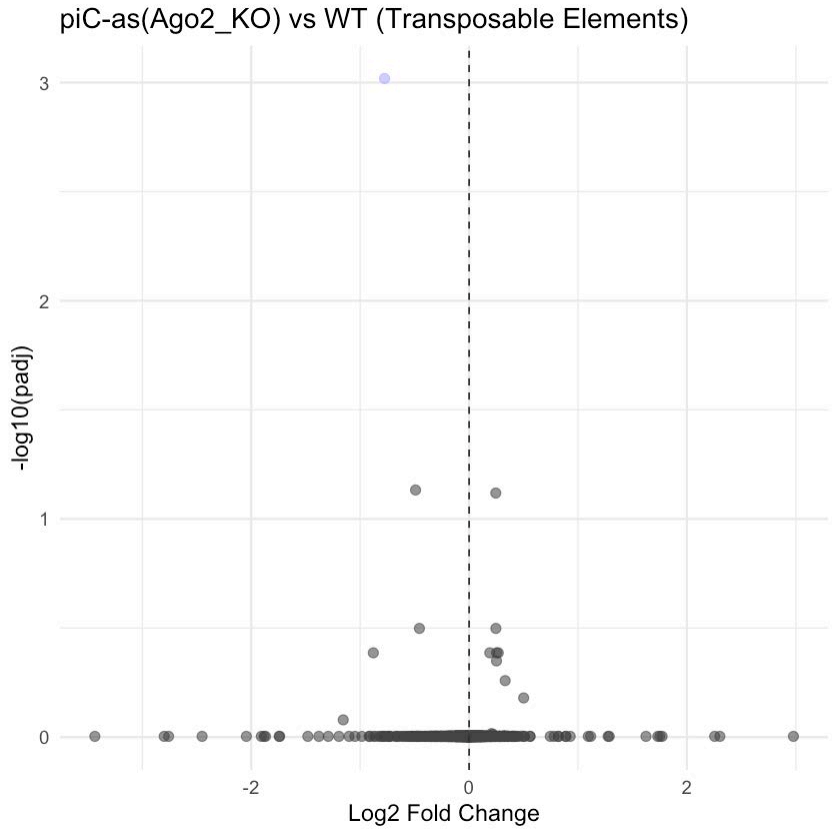
Analyze TE expression in pic-as(Ago2)_Fig. 2c

```
################################################################################
# Title: Differential Expression Analysis of TEs in pic-as(Ago2) Knockout Testes
# Description: This script performs DESeq2 analysis of transposable elements in RNA-seq
#              data from WT and pic-as(Ago2) knockout mouse testes and generates a
#              volcano plot of differential expression results
################################################################################


#     Load count matrices   #


base_dir <- "/Users/loubalovaz2/Documents/Lab/Pachytene piRNA
project/Library_prep/240220_Spin1, Ago2, WT - RNAseq_whole
testis/Long_RNA_libraries/Proper_50_PE_seq/featureCounts_TE"

count_files <- c(
  file.path(base_dir, "161212_Zuzana-
7repeat_S2_L008_R1_001.fastq.gz.mapped.Aligned.sortedByCoord.out.bam_featureCounts_TE.tx
t"),
  file.path(base_dir,
"161213_s08_S6_R1_001.fastq.gz.mapped.Aligned.sortedByCoord.out.bam_featureCounts_TE.txt
"),
  file.path(base_dir,
"161922_s09_S7_R1_001.fastq.gz.mapped.Aligned.sortedByCoord.out.bam_featureCounts_TE.txt
"),
  file.path(base_dir,
"1069_s04_S3_R1_001.fastq.gz.mapped.Aligned.sortedByCoord.out.bam_featureCounts_TE.txt")
```

```
,
  file.path(base_dir,
"1070_s05_S4_R1_001.fastq.gz.mapped.Aligned.sortedByCoord.out.bam_featureCounts_TE.txt")
,
  file.path(base_dir,
"1071_s06_S5_R1_001.fastq.gz.mapped.Aligned.sortedByCoord.out.bam_featureCounts_TE.txt")
)

# Read in and combine count files
count_list <- lapply(count_files, function(file) {
  read.table(file, header = TRUE, row.names = 1)
})

count_matrix_all <- do.call(cbind, count_list)


#     Select count columns  #


count_matrix <- count_matrix_all[, c(6, 12, 18, 24, 30, 36)]
colnames(count_matrix) <- c("WT_1", "WT_2", "WT_3", "Ago2_1", "Ago2_2", "Ago2_3")

# Convert fractional read counts to whole numbers
count_matrix <- round(count_matrix)


#       Sample metadata     #


sample_metadata <- data.frame(
  row.names = colnames(count_matrix),
  Genotype = c("WT", "WT", "WT", "Ago2_KO", "Ago2_KO", "Ago2_KO")
)

# Verify that sample names match between counts and metadata
stopifnot(all(colnames(count_matrix) %in% rownames(sample_metadata)))
stopifnot(all(colnames(count_matrix) == rownames(sample_metadata)))


#      DeSeq2 analysis      #


dds <- DESeqDataSetFromMatrix(
  countData = count_matrix,
  colData = sample_metadata,
  design = ~ Genotype
)

# Set reference level
dds$Genotype <- relevel(dds$Genotype, ref = "WT")

# Run DESeq2
dds <- DESeq(dds)
res <- results(dds)


#       Save results        #


res$gene_id <- rownames(res)
```

```r
res_df <- as.data.frame(res)

write_xlsx(res_df, file.path(output_dir, "Ago2_TE_DESeq2_results.xlsx"))


#       Volcano plot        #


res_df$color <- "#484545"
res_df$color[res_df$padj < 0.05 & res_df$log2FoldChange > 0] <- "#f80000"
res_df$color[res_df$padj < 0.05 & res_df$log2FoldChange < 0] <- "#aeaefc"

volcano_plot_Ago2_TE <- ggplot(res_df, aes(x = log2FoldChange, y = -log10(padj), color =
color)) +
  geom_point(alpha = 0.6, size = 2.5) +
  geom_vline(xintercept = 0, linetype = "dashed", color = "#343333") +
  scale_color_identity() +
  labs(
    x = "Log2 Fold Change",
    y = "-log10(padj)",
    title = "piC-as(Ago2_KO) vs WT (Transposable Elements)"
  ) +
  theme_minimal(base_size = 14)

ggsave(file.path(output_dir, "Ago2_TE_volcano.svg"), plot = volcano_plot_Ago2_TE, bg =
"white", width = 6, height = 5)
ggsave(file.path(output_dir, "Ago2_TE_volcano.png"), plot = volcano_plot_Ago2_TE, bg =
"white", width = 6, height = 5)
```

```r
print(volcano_plot_Ago2_TE)
```

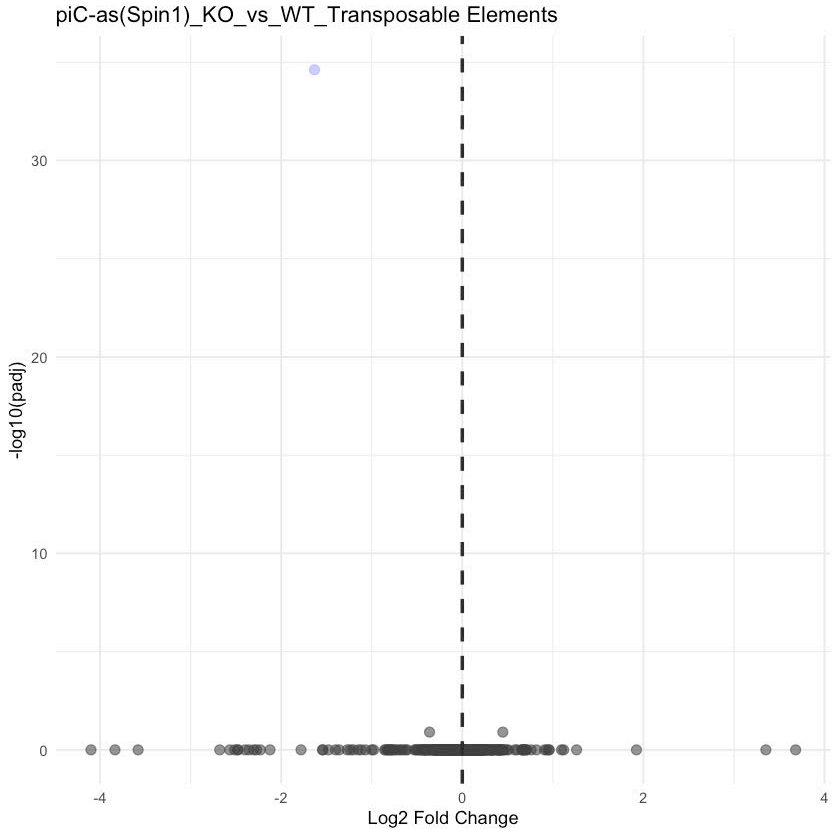


Check if there are any deregulated TEs in piC-as(Spin1)_Extended Fig. 3f

```r
################################################################################
# Title: Differential Expression Analysis of TEs in pic-as(Spin1) Knockout Testes
# Description: This script performs DESeq2 analysis of transposable elements in RNA-seq
#              data from WT and pic-as(Spin1) knockout mouse testes and generates a
#              volcano plot of differential expression results
################################################################################


#     Load count matrices   #


base_dir <- "/Users/loubalovaz2/Documents/Lab/Pachytene piRNA
project/Library_prep/240220_Spin1, Ago2, WT - RNAseq_whole
testis/Long_RNA_libraries/Proper_50_PE_seq/featureCounts_TE"

count_files_TE <- c(
  file.path(base_dir, "161212_Zuzana-
7repeat_S2_L008_R1_001.fastq.gz.mapped.Aligned.sortedByCoord.out.bam_featureCounts_TE.tx
t"),
  file.path(base_dir,
"161213_s08_S6_R1_001.fastq.gz.mapped.Aligned.sortedByCoord.out.bam_featureCounts_TE.txt
"),
  file.path(base_dir,
"161922_s09_S7_R1_001.fastq.gz.mapped.Aligned.sortedByCoord.out.bam_featureCounts_TE.txt
"),
  file.path(base_dir, "161602_Zuzana-
1repeat_S1_L008_R1_001.fastq.gz.mapped.Aligned.sortedByCoord.out.bam_featureCounts_TE.tx
t"),
  file.path(base_dir,
"161603_s02_S1_R1_001.fastq.gz.mapped.Aligned.sortedByCoord.out.bam_featureCounts_TE.txt
"),
  file.path(base_dir,
"161604_s03_S2_R1_001.fastq.gz.mapped.Aligned.sortedByCoord.out.bam_featureCounts_TE.txt
")
)

# Read and combine count files into a single count matrix
count_matrix_list_TE <- lapply(count_files_TE, function(file) {
  read.table(file, header = TRUE, row.names = 1)
})
count_matrix_full <- do.call(cbind, count_matrix_list_TE)


#     Select count columns  #


count_matrix <- count_matrix_full[, c(6, 12, 18, 24, 30, 36)]

# Rename columns
colnames(count_matrix) <- c("WT_1", "WT_2", "WT_3", "Spin1_1", "Spin1_2", "Spin1_3")


#       Sample metadata     #


sample_metadata <- data.frame(
  row.names = c("WT_1", "WT_2", "WT_3", "Spin1_1", "Spin1_2", "Spin1_3"),
  Genotype = c("WT", "WT", "WT", "Spin1_KO", "Spin1_KO", "Spin1_KO")
)
```

```r
# Verify that sample names match between counts and metadata
stopifnot(all(colnames(count_matrix) %in% rownames(sample_metadata)))
stopifnot(all(colnames(count_matrix) == rownames(sample_metadata)))

# Convert fractional counts to whole numbers
count_matrix <- round(count_matrix)


#       DeSeq2 analysis     #


# Create DESeq2 dataset
dds <- DESeqDataSetFromMatrix(
  countData = count_matrix,
  colData = sample_metadata,
  design = ~ Genotype
)

# Set WT as the reference level
dds$Genotype <- relevel(dds$Genotype, ref = "WT")

# Run DESeq2
dds <- DESeq(dds)
res <- results(dds)


#       Save results        #


res$gene_id <- rownames(res)
res.df <- as.data.frame(res)

write_xlsx(res.df, file.path(output_dir, "Spin1_TE_DESeq2_results.xlsx"))


#       Volcano plot        #


# Prepare data for volcano plot
res.df$color <- "#484545"
res.df$color[res.df$padj < 0.05 & res.df$log2FoldChange > 0] <- "#f80000"
res.df$color[res.df$padj < 0.05 & res.df$log2FoldChange < 0] <- "#aeaefc"

Spin1_TE_volcano <- ggplot(res.df, aes(x = log2FoldChange, y = -log10(padj), color =
color)) +
  geom_point(alpha = 0.6, size = 2.5) +
  geom_vline(xintercept = 0, color = "#343333", linetype = "dashed", linewidth = 1) +
  scale_color_identity() +
  labs(x = "Log2 Fold Change", y = "-log10(padj)", title = "piC-
as(Spin1)_KO_vs_WT_Transposable Elements") +
  theme_minimal()

# Save plots
ggsave(file.path(output_dir, "Spin1_TE_volcano.svg"), bg = "white", plot =
Spin1_TE_volcano, device = "svg")
ggsave(file.path(output_dir, "Spin1_TE_volcano.png"), bg = "white", plot =
Spin1_TE_volcano, device = "png")
```

```
print(Spin1_TE_volcano)
```



Check if the deletion of pic-as(Ago2) promoter affected expression of any other piRNA precursor_Fig. 2d

```
#     piRNA precursors      #

#Load library
library(readxl)
#Create gtf file with 90th percentile of top productive MIWI piRNA clusters
(Konstantinidou, Loubalova, et al., Cell Reports, 2024)
#load MIWI_clusters from Cell Reports publication
MIWI_clusters_xslx <- read_xlsx("/Users/loubalovaz2/Documents/Lab/Pachytene piRNA
project/Library_prep/240220_Spin1, Ago2, WT - RNAseq_whole
testis/Long_RNA_libraries/Old_Seq/Run_Samples_2,3,4,5,6,8,9,/240423_Miwi-PICB_pachytene-
short.xlsx")
#save as data frame
MIWI_clusters <- as.data.frame(MIWI_clusters_xslx)
#arrange by all reads explained=forced mapping for productivity
MIWI_clusters_ordered <-
MIWI_clusters[order(MIWI_clusters$C11_rank_by_all_reads_explained), ]
#take top 64 that the publication defined as 90th percentile
MIWI_clusters_90thpercentile <- MIWI_clusters_ordered[1:64, ]
#give them names (numbers) based on productivity
MIWI_clusters_90thpercentile$gene_id <-
MIWI_clusters_90thpercentile$C11_rank_by_all_reads_explained
MIWI_clusters_90thpercentile$transcript_id <- MIWI_clusters_90thpercentile$gene_id
MIWI_clusters_90thpercentile$chromosome <- MIWI_clusters_90thpercentile$seqnames
MIWI_clusters_90thpercentile$feature_type <- "gene"
```

```
#     Generate GTF file     #


# Function to convert data frame to GTF format
convert_to_gtf <- function(df) {
  gtf <- paste(
    df$chromosome,
    "source",
    df$feature_type,
    df$start,
    df$end,
    ".",
    df$strand,
    ".",
    paste("gene_id", df$gene_id, sep = " "),
    paste("transcript_id", df$transcript_id, sep = " "),
    sep = "\t"
  )
  return(gtf)
}

# Convert data frame to GTF format
gtf_MIWI_clusters_90thpercentile <- convert_to_gtf(MIWI_clusters_90thpercentile)


#       Save GTF file       #

# Write GTF content to a file and use it for FeatureCounts
writeLines(gtf_MIWI_clusters_90thpercentile, file.path(output_dir,
"gtf_MIWI_clusters_90thpercentile.gtf"))
```

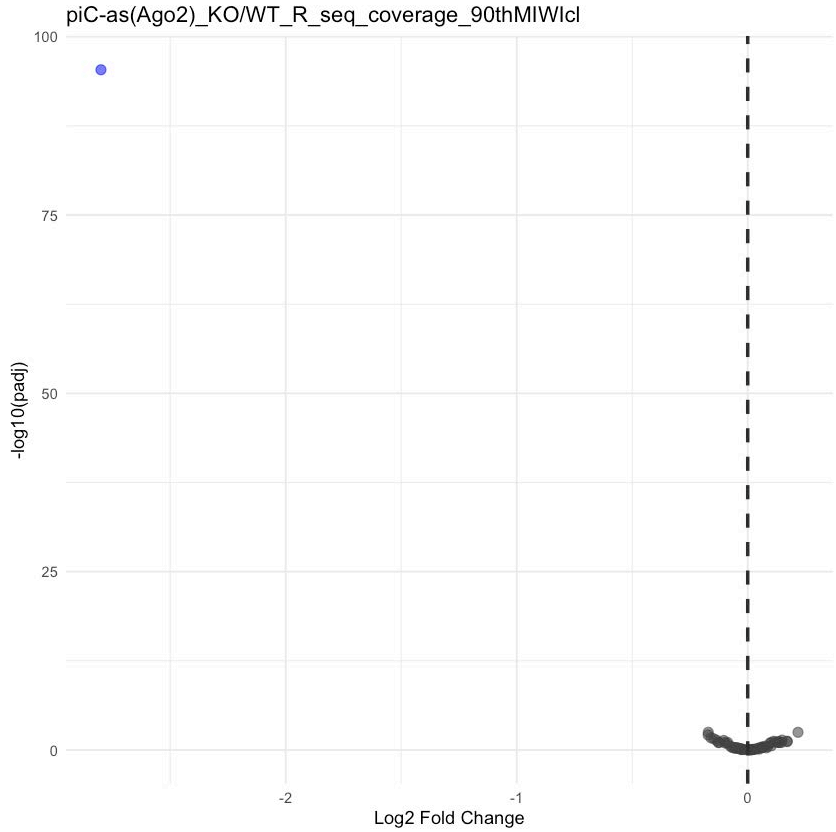
```
# BASH - Use FeatureCounts to define the coverage of MIWI piRNA precursors in RNA seq of
piC-as(Ago2) and WT mice using the gtf_MIWI_clusters_90thpercentile GTF file

#!/bin/bash
#FeatureCounts for piRNA precursor expression

module load subread

for run in  161212_Zuzana-7repeat_S2_L008_R1_001 161213_s08_S6_R1_001
161922_s09_S7_R1_001 1069_s04_S3_R1_001 1070_s05_S4_R1_001  1071_s06_S5_R1_001 ; do
input_file1="${run}.fastq.gz.mapped.Aligned.sortedByCoord.out.bam"
output_prefix1="${run}_featureCount_whole_testis_RNA_allMIWI_clusters"

featureCounts \
-p \
--countReadPairs \
-s 2 \
-M \
--fraction \
-t gene \
-a /path/gtf_MIWI_clusters_90thpercentile.gtf \
-o /path/FeatureCounts_90thpercentile_MIWI_clusters/"${output_prefix1}.txt"
$input_file1;

echo "    ✓ Finished ${run}"
done
```

```r
################################################################################
# Title: Differential Expression Analysis of piRNA precursors in pic-as(Ago2) Knockout
Testes
# Description: This script performs DESeq2 analysis of piRNA precursors (90th percentile
of top productive MIWI piRNA precursors)
#              in RNA-seq data from WT and pic-as(Ago2) knockout mouse testes and
generates a
#               volcano plot of differential expression results
################################################################################


#     Load count matrices   #


base_dir_WT <- "/Users/loubalovaz2/Documents/Lab/Writing/Pachytene_clusters
paper/statistics/piRNA/whole_testis/FeatureCounts_90thpercentile_MIWI_clusters"
base_dir_KO <- "/Users/loubalovaz2/Documents/Lab/Writing/Pachytene_clusters
paper/250922/Ago2/featureCounts_MIWIcl_90th_percentile_RNA_coverage"

count_files_Ago2_whole_testis_90th <- c(
  file.path(base_dir_WT, "161212_Zuzana-
7repeat_S2_L008_R1_001_featureCount_whole_testis_RNA_allMIWI_clusters.txt"),
  file.path(base_dir_WT,
"161213_s08_S6_R1_001_featureCount_whole_testis_RNA_allMIWI_clusters.txt"),
  file.path(base_dir_WT,
"161922_s09_S7_R1_001_featureCount_whole_testis_RNA_allMIWI_clusters.txt"),
  file.path(base_dir_KO,
"1069_s04_S3_R1_001_featureCount_whole_testis_RNA_allMIWI_clusters.txt"),
  file.path(base_dir_KO,
"1070_s05_S4_R1_001_featureCount_whole_testis_RNA_allMIWI_clusters.txt"),
  file.path(base_dir_KO,
"1071_s06_S5_R1_001_featureCount_whole_testis_RNA_allMIWI_clusters.txt")
)

# Read in and combine count files
count_matrix_list_Ago2_whole_testis_90th <- lapply(count_files_Ago2_whole_testis_90th,
function(file) {
  count_data <- read.table(file, header = TRUE, row.names = 1)
  return(count_data)
})

# Combine the count matrices into a single matrix
count_matrix1_Ago2_whole_testis_90th <- do.call(cbind,
count_matrix_list_Ago2_whole_testis_90th)


#     Select count columns  #


count_matrix1_Ago2_whole_testis_90th <- count_matrix1_Ago2_whole_testis_90th[, c(6, 12,
18, 24, 30, 36)]
new_column_names_Ago2_whole_testis_90th <- c("WT_1", "WT_2", "WT_3", "Ago2_1", "Ago2_2",
"Ago2_3")
colnames(count_matrix1_Ago2_whole_testis_90th) <-
new_column_names_Ago2_whole_testis_90th

# Convert fractional read counts to whole numbers
count_matrix1_Ago2_whole_testis_90th <- round(count_matrix1_Ago2_whole_testis_90th)
```

```
#       Sample metadata     #


# Create a sample metadata data frame
sample_metadata_Ago2_whole_testis_90th <- data.frame(
  row.names = c("WT_1", "WT_2", "WT_3", "Ago2_1", "Ago2_2", "Ago2_3"), Genotype =
c("WT", "WT", "WT","Ago2_KO", "Ago2_KO", "Ago2_KO"))

# Verify that sample names match between counts and metadata
stopifnot(all(colnames(count_matrix1_Ago2_whole_testis_90th) %in%
rownames(sample_metadata_Ago2_whole_testis_90th)))
stopifnot(all(colnames(count_matrix1_Ago2_whole_testis_90th) ==
rownames(sample_metadata_Ago2_whole_testis_90th)))


#       DeSeq2 analysis     #


# Create a DESeqDataSet
dds_Ago2_whole_testis_90th <- DESeqDataSetFromMatrix(countData =
count_matrix1_Ago2_whole_testis_90th,
                                          colData =
sample_metadata_Ago2_whole_testis_90th,
                                          design = ~ Genotype)

# Set reference level
dds_Ago2_whole_testis_90th$Genotype <- relevel(dds_Ago2_whole_testis_90th$Genotype, ref
= "WT")


# Run DESeq2
dds_Ago2_whole_testis_90th <- DESeq(dds_Ago2_whole_testis_90th)
res_Ago2_whole_testis_90th <- results(dds_Ago2_whole_testis_90th)


#       Save results        #


res_Ago2_whole_testis.df_90th <- as.data.frame(res_Ago2_whole_testis_90th)
class(res_Ago2_whole_testis_90th)

res.df.1_Ago2_whole_testis_90th <- res_Ago2_whole_testis.df_90th
res.df.1_Ago2_whole_testis_90th$gene_id <- rownames(res.df.1_Ago2_whole_testis_90th)
write_xlsx(res.df.1_Ago2_whole_testis_90th, file.path(output_dir,
"Ago2_R_seq_90thMIWIclusters_coveraege_VolcanoPlot.xlsx"))


#       Volcano plot        #


res.df.1_Ago2_whole_testis_90th$color <- "#484545"
res.df.1_Ago2_whole_testis_90th$color[res.df.1_Ago2_whole_testis_90th$padj < 0.05 &
res.df.1_Ago2_whole_testis_90th$log2FoldChange > 1] <- "#f80000"
res.df.1_Ago2_whole_testis_90th$color[res.df.1_Ago2_whole_testis_90th$padj < 0.05 &
res.df.1_Ago2_whole_testis_90th$log2FoldChange < -1] <- "#0404ef"

Ago2_R_seq_90thMIWIcl_volcano <- ggplot(res.df.1_Ago2_whole_testis_90th, aes(x =
log2FoldChange, y = -log10(padj), color = color)) +
```

```
  geom_point(alpha = 0.6, size = 2.5) +
  geom_vline(xintercept = 0,
             color = "#343333",
             linetype = "dashed",
             linewidth = 1) +
  scale_color_identity() +
  labs(x = "Log2 Fold Change", y = "-log10(padj)", title = "piC-
as(Ago2)_KO/WT_R_seq_coverage_90thMIWIcl") +
  theme_minimal()

ggsave(file.path(output_dir, "Ago2_R_seq_90thMIWIcl_volcano_lfc>1.svg"), bg = "white",
plot = Ago2_R_seq_90thMIWIcl_volcano, device = "svg")
ggsave(file.path(output_dir, "Ago2_R_seq_90thMIWIcl_volcano_lfc>1.png"), bg = "white",
plot = Ago2_R_seq_90thMIWIcl_volcano, device = "png")
```

```
print(Ago2_R_seq_90thMIWIcl_volcano)
```



Check if the deletion of piC-as(Ago2) promoter affected production of piRNAs from any other cluster_Fig. 2d

```
# BASH - Use FeatureCounts to define the coverage of MIWI piRNA clusters in smallRNA seq
of piC-as(Ago2) and WT mice using the gtf_MIWI_clusters_90thpercentile GTF file

#  - PROCESSING AND MAPPING OF SMALL RNA SEQ DATA IS REPORTED IN CODE FOR MOUSE_FIGURE1

#!/bin/bash

module load subread

for run in NonStructural_Mouse_161212_testes_small_RNAs_ZL5_S6
NonStructural_Mouse_161922_testes_small_RNAs_ZL6_S7
```

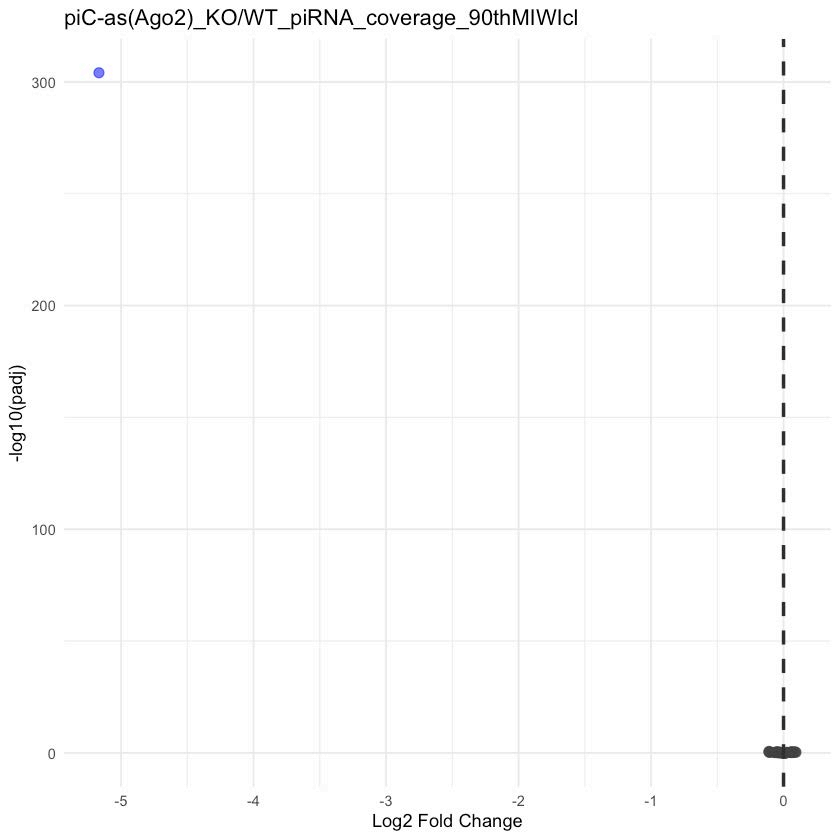
```
NonStructural_Mouse_1069_testes_small_RNAs_ZL3_S4
NonStructural_Mouse_1070_testes_small_RNAs_ZL4_S5; do
input_file1="${run}_R1_trimmed_collapsed_noUMI_nomiRNA_mm10Aligned-
STAR_sortedByCoord.bam"
output_prefix1="${run}_featureCount"

featureCounts \
-s 1 \
-M \
-O \
--fraction \
-t gene \
-a /path/gtf_MIWI_clusters_90thpercentile.gtf \
-o /path/FeatureCounts_90thpercentile/"${output_prefix1}.txt" \
$input_file1;

echo "    ✓ Finished ${run}"
done
```

```
################################################################################
# Title: Differential Expression Analysis of piRNA clusters in pic-as(Ago2) Knockout
Testes
# Description: This script performs DESeq2 analysis of piRNAs (90th percentile of top
productive MIWI piRNA clusters)
#              in small RNA-seq data from WT and pic-as(Ago2) knockout mouse testes and
generates a
#               volcano plot of differential expression results
################################################################################


#     Load count matrices   #

# List of count files
base_dir_WT <- "/Users/loubalovaz2/Documents/Lab/Writing/Pachytene_clusters
paper/statistics/piRNA/FeatureCounts_90thpercentile"
base_dir_KO <- "/Users/loubalovaz2/Documents/Lab/Writing/Pachytene_clusters
paper/250922/Ago2/featureCounts_MIWI_90th_percentile"

count_files_Ago2_small_90th <- c(
  file.path(base_dir_WT,
"NonStructural_Mouse_161212_testes_small_RNAs_ZL5_S6_featureCount.txt"),
  file.path(base_dir_WT,
"NonStructural_Mouse_161922_testes_small_RNAs_ZL6_S7_featureCount.txt"),
  file.path(base_dir_KO,
"NonStructural_Mouse_1069_testes_small_RNAs_ZL3_S4_featureCount.txt"),
  file.path(base_dir_KO,
"NonStructural_Mouse_1070_testes_small_RNAs_ZL4_S5_featureCount.txt")
)

# Read in and combine count files into a single count matrix
count_matrix_list_Ago2_small_90th <- lapply(count_files_Ago2_small_90th, function(file)
{
  count_data <- read.table(file, header = TRUE, row.names = 1)
  return(count_data)
})

# Combine the count matrices into a single matrix
count_matrix1_Ago2_small_90th <- do.call(cbind, count_matrix_list_Ago2_small_90th)
```

```
#     Select count columns  #

#choose only columns with gene count for given sample
count_matrix_Ago2_small_90th <- count_matrix1_Ago2_small_90th[, c(6, 12, 18, 24)]
new_column_names_Ago2_small_90th <- c("WT_1", "WT_2", "Ago2_1", "Ago2_2")
colnames(count_matrix_Ago2_small_90th) <- new_column_names_Ago2_small_90th

# Convert fractional read counts to whole numbers
count_matrix_Ago2_small_90th <- round(count_matrix_Ago2_small_90th)


#       Sample metadata     #


# Create a sample metadata data frame
sample_metadata_Ago2_small_90th <- data.frame(
  row.names = c("WT_1", "WT_2", "Ago2_1", "Ago2_2"), Genotype = c("WT", "WT", "Ago2_KO",
"Ago2_KO"))

# Verify that sample names match between counts and metadata
stopifnot(all(colnames(count_matrix_Ago2_small_90th) %in%
rownames(sample_metadata_Ago2_small_90th)))
stopifnot(all(colnames(count_matrix_Ago2_small_90th) ==
rownames(sample_metadata_Ago2_small_90th)))


#       DeSeq2 analysis     #


# Create a DESeqDataSet
dds_Ago2_small_90th <- DESeqDataSetFromMatrix(countData = count_matrix_Ago2_small_90th,
                                              colData = sample_metadata_Ago2_small_90th,
                                              design = ~ Genotype)

# Set the factor level
dds_Ago2_small_90th$Genotype <- relevel(dds_Ago2_small_90th$Genotype, ref = "WT")


#Run DESeq2
dds_Ago2_small_90th <- DESeq(dds_Ago2_small_90th)
res_Ago2_small_90th <- results(dds_Ago2_small_90th)


#       Save results        #


#save results as data.frame
res_Ago2_small.df_90th <- as.data.frame(res_Ago2_small_90th)
class(res_Ago2_small.df_90th)

res.df.1_Ago2_small_90th <- res_Ago2_small.df_90th
res.df.1_Ago2_small_90th$gene_id <- rownames(res.df.1_Ago2_small_90th)
write_xlsx(res.df.1_Ago2_small_90th,
file.path(output_dir,"Ago2_piRNA_90thMIWIclusters_coveraege_VolcanoPlot.xlsx"))


#       Volcano plot        #
```

```
res.df.1_Ago2_small_90th$color <- "#484545"
res.df.1_Ago2_small_90th$color[res.df.1_Ago2_small_90th$padj < 0.05 &
res.df.1_Ago2_small_90th$log2FoldChange > 1] <- "#f80000"
res.df.1_Ago2_small_90th$color[res.df.1_Ago2_small_90th$padj < 0.05 &
res.df.1_Ago2_small_90th$log2FoldChange < -1] <- "#0404ef"

Ago2_piRNA_90thMIWIcl_volcano <- ggplot(res.df.1_Ago2_small_90th, aes(x =
log2FoldChange, y = -log10(padj), color = color)) +
  geom_point(alpha = 0.6, size = 2.5) +
  geom_vline(xintercept = 0,
             color = "#343333",
             linetype = "dashed",
             linewidth = 1) +
  scale_color_identity() +
  labs(x = "Log2 Fold Change", y = "-log10(padj)", title = "piC-
as(Ago2)_KO/WT_piRNA_coverage_90thMIWIcl") +
  theme_minimal()

ggsave(file.path(output_dir, "Ago2_piRNA_90thMIWIcl_volcano.svg"), bg = "white", plot =
Ago2_piRNA_90thMIWIcl_volcano, device = "svg")
ggsave(file.path(output_dir, "Ago2_piRNA_90thMIWIcl_volcano.png"), bg = "white", plot =
Ago2_piRNA_90thMIWIcl_volcano, device = "png")
```

```
print(Ago2_piRNA_90thMIWIcl_volcano)
```



Sequence logo for piC-as(Ago2)-derived piRNAs

```
suppressPackageStartupMessages({
    library(GenomicRanges)
    library(PICB)
    library(BSgenome.Mmusculus.UCSC.mm10)
```

```
    library(GenomicAlignments)
    library(ggplot2)
    library(ggseqlogo)
    library(patchwork)
})
source("../scripts/plotLogo.R")
options(repr.plot.width=10, repr.plot.height=5)
```

```
#piRNA cluster coordinates from Konstantinidou et al. (2024) in Cell Reports and rank
load("../data/annotations/MILIclusters_pachytene.RData")
MILIclusters_pach <- MILIclusters$clusters

MILIclusters_pach <- MILIclusters_pach[order(-
MILIclusters_pach$all_reads_primary_alignments_FPM), ]
MILIclusters_pach$rankByAllReadsPrimaryAlignmentsFPM <- seq_along(MILIclusters_pach)
MILIclusters_pach <- keepStandardChromosomes(MILIclusters_pach)
paste0("Number of piRNA clusters in MILI pachytene by PICB: ",
length(MILIclusters_pach))
```

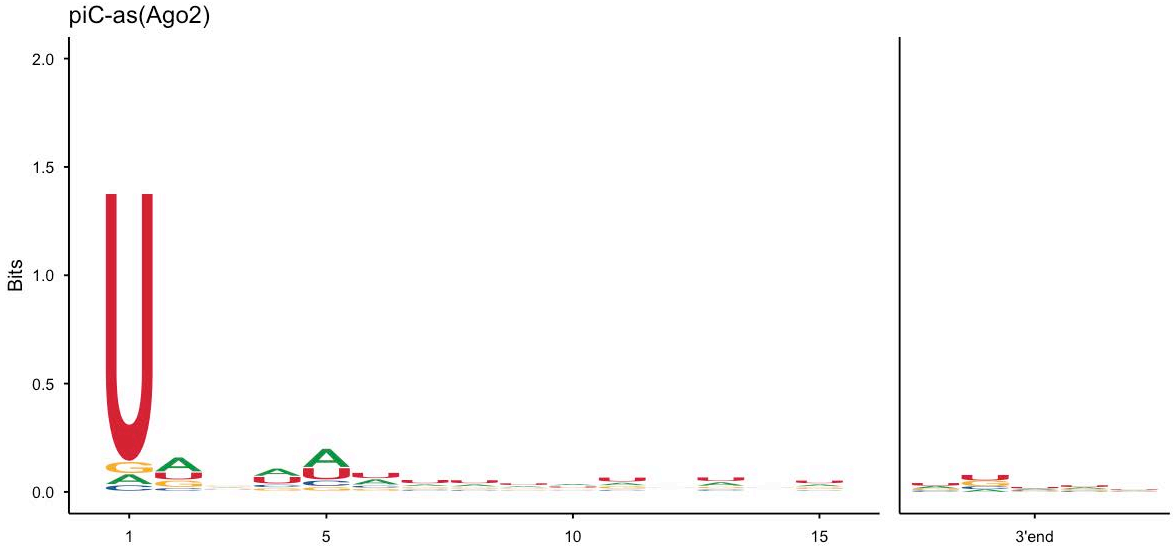
'Number of piRNA clusters in MILI pachytene by PICB: 4188'

```
# load alignments to mm10 genome
gr_mm10 <- PICBload(
    BAMFILE =
"../data/bam/Mouse_161922_testes_small_RNAs_ZL6_S7_R1_trimmed_UMIcollapsed_structOut_miR
outwS_Aligned.sortedByCoord.out.bam",
    REFERENCE.GENOME = "BSgenome.Mmusculus.UCSC.mm10",
    GET.ORIGINAL.SEQUENCE = TRUE, VERBOSE = FALSE, WHAT = c("flag", "cigar"))

gr_mm10 <- gr_mm10$unique
gr_mm10 <- keepStandardChromosomes(gr_mm10)
```

```
# Load BAM file of piRNAs targeting protein coding genes (in custom transcriptome)
bamPCG_dir <-
"../data/bam/mmuToTranscriptome/PCG_Mouse_161922_testes_small_RNAs_ZL6_S7_R1_trimmed_UMI
collapsed_structOut_miRoutwS_Aligned.PICBloadWseqs.primAlignWinfo..PCGtranscriptome_clip
5pNbases1_Extend5pOfRead1_minMatch19_Aligned.sortedByCoord.out.bam"

bam <- BamFile(bamPCG_dir)
# exclude both secondary alignments and supplementary alignments
fields <- scanBamWhat()
primary_flag <- scanBamFlag(isSecondary = FALSE, isSupplementary = FALSE)
param <- ScanBamParam(flag = primary_flag,
    what=c('qname','flag','rname','strand','pos','qwidth', 'cigar', 'seq'),
    tag=c('NH', "MD"))

ga_all_alignments <- readGAlignments(bam, param = param)
PCG_total_reads <- length(ga_all_alignments)
ga_alignments <- ga_all_alignments[mcols(ga_all_alignments)$NH == 1] #filter by unique
alignments
PCG_unique_reads <- length(ga_alignments)

gr_alignments <- makeGRangesFromDataFrame(as.data.frame(ga_alignments),
keep.extra.columns=TRUE)

mcols(gr_alignments) <- mcols(gr_alignments)[ , !(colnames(mcols(gr_alignments)) %in%
c("njunc", "strandInfo", "rname", "strand.1", "pos", "qwidth.1", "cigar.1", "qual"))]
```

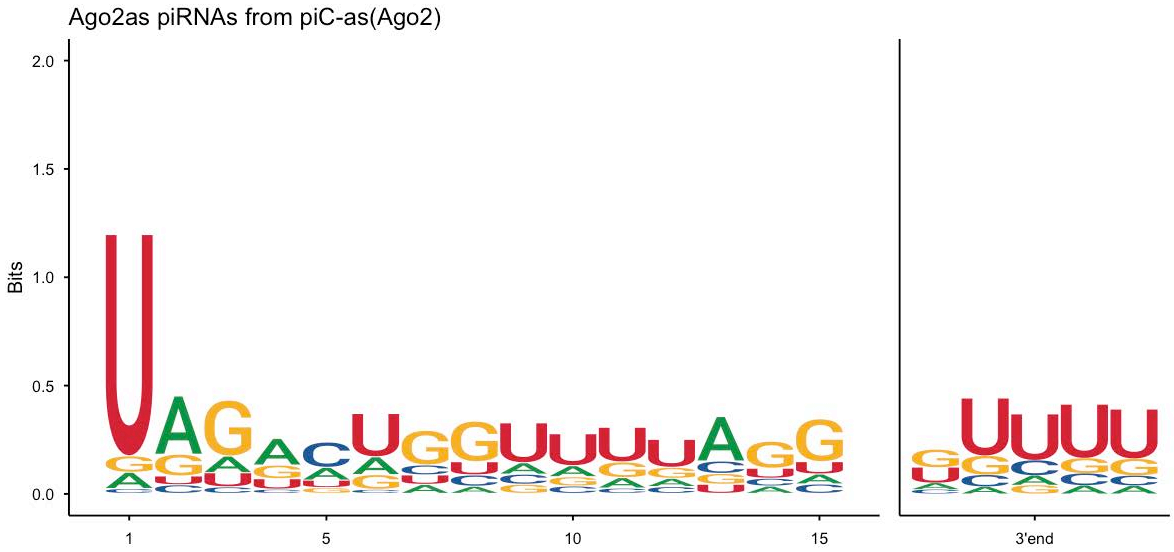
```
# Sequence logo of piC-as(Ago2)-derived piRNAs: total piRNAs - Fig 2e
piCasAgo2_logo <- logoPlot(subsetByOverlaps(gr_mm10, MILIclusters_pach[56]), "piC-
as(Ago2)", genome = BSgenome.Mmusculus.UCSC.mm10)
piCasAgo2_logo
```

```
Scale for x is already present.
Adding another scale for x, which will replace the existing scale.
[1] "Using provided BSgenome object to build 3' extension logo."
Scale for x is already present.
Adding another scale for x, which will replace the existing scale.
```




```
# Sequence logo of piC-as(Ago2)-derived piRNAs: piRNAs targeting Ago2 - Extended Data
Fig 2d
# Get all the readnames of piRNAs targeting Ago2 (coordinates in PCG transcriptome:
chr15:1357924-1365954:-)
Ago2as_piRNAs <- gr_mm10[names(gr_mm10) %in% sub("_.*", "",
subsetByOverlaps(gr_alignments, invertStrand(GRanges("chr15:1357924-1365954:-
")))$qname)]
# Subset by piC-as(Ago2)
Ago2as_piRNAsFromPiCasAgo2 <- subsetByOverlaps(Ago2as_piRNAs, MILIclusters_pach[56])
# Sequence logo
Ago2as_piRNAsFromPiCasAgo2_logo <- logoPlot(Ago2as_piRNAsFromPiCasAgo2, "Ago2as piRNAs
from piC-as(Ago2)", genome = BSgenome.Mmusculus.UCSC.mm10)
Ago2as_piRNAsFromPiCasAgo2_logo
```

```
Scale for x is already present.
Adding another scale for x, which will replace the existing scale.
[1] "Using provided BSgenome object to build 3' extension logo."
Scale for x is already present.
Adding another scale for x, which will replace the existing scale.
```


Ago2as piRNAs from piC-as(Ago2)
Bits
2.0
1.5
1.0
0.5
0.0
1
5
10
15
3'end

Human Adult piRNA Gene Targeting and their Conservation across Mammals

Code for: Manuscript Figure 4 and Extended Data Figure 4

Introduction to file

Jupyter Notebook used R/4.3 and occasionally Bash.

Human adult sample from SRR8575350 published in Özata, et al. (2020, Nat Ecol Evol).

In this document, piCs and their corresponding rank are derived from adult piCs published in Konstantinidou, et al. (2024, Cell Reports). piCs were created with the Bioconductor package PICB.
For code purposes (e.g. for relating piRNAs to specific piCs) piRNAs that are not from piCs have rank 0.

GOLGA2-targeting
piC-as(GOLGA2) - rank: 1, chr15:62207201-62280100 (+)

Processing small RNA-seq

Preparation: PCG-Transcriptome

Protein-coding gene transcriptome creation with ../scripts/buildTranscriptomes.R

Mapping all piRNAs to PCG-transcriptome

Create fasta with all mapped reads including rank, coordinates of mapping and human piRNA group

```
suppressPackageStartupMessages({
    library("GenomicRanges")
    library("Biostrings")
})
```

```
align_69yo <-
readRDS("../../../OneDrive/3_hsa_detection_prepach_adult/Code/STAROutput/oxSmallRNASeq/smallRNA_hsa_69y_A01_SRR8575350/smallRNA_hsa_69y_A01_SRR8575350_cleaned_trimmed_snoMiTRout_20to40nt_Aligned.PICBloadWithSeq.rds")

align_69yo_allPrimary <- c(align_69yo$unique, align_69yo$multi.primary)
length(align_69yo_allPrimary)
```

39486612

```
#get cluster coordinates (ranked!) from Cell Reports publication (human 69 yo)
clusters_69yo <-
readRDS("../../../OneDrive/3_hsa_detection_prepach_adult/Code/STAROutput/oxSmallRNASeq/smallRNA_hsa_69y_A01_SRR8575350/clusters.ranked.smallRNA_hsa_69y_A01_SRR8575350_cleaned_trimmed_snoMiTRout_20to40nt_Aligned.sortedByCoord.ForceMapped.RDS")
```

```
clusters_69yo <- clusters_69yo[order(clusters_69yo$rank_readsExplained), ]
clusters_69yo$rank_readsExplained <- seq_along(clusters_69yo)
length(clusters_69yo)
```

13098

```
# load disentanglement_piCs_A01_69yo_SRR8575350_FM_90thPercentile_v2.RDS from
STAROutput/oxSmallRNASeq/smallRNA_hsa_69y_A01_SRR8575350/
piCs_A01_69yo_disentangled <-
readRDS("../../../OneDrive/3_hsa_detection_prepach_adult/Code/disentangledCluster/hsa_di
sentanglement_piCs_A01_69yo_SRR8575350_FM_90thPercentile_v2.RDS")

#add to readname rank_chr_startPos_strand_NH_piCgroup
#piCs_A01_69yo_disentangled is in three groups: unique69yo, commonShort, commonLong, add
this information (unique69yo="UN", commonSSC="CS", commonDynamic="CD") to another column
in clusters_69yo based on their shared column rank_readsExplained. If
rank_readsExplained is not listed in piCs_A01_69yo_disentangled, add "NA".
clusters_69yo$piCgroup <- "NA"
clusters_69yo$piCgroup[clusters_69yo$rank_readsExplained %in%
piCs_A01_69yo_disentangled$unique69yo$rank_readsExplained] <- "UN"
clusters_69yo$piCgroup[clusters_69yo$rank_readsExplained %in%
piCs_A01_69yo_disentangled$commonShort$rank_readsExplained] <- "CS"
clusters_69yo$piCgroup[clusters_69yo$rank_readsExplained %in%
piCs_A01_69yo_disentangled$commonLong$rank_readsExplained] <- "CD"
```

```
# Find overlaps between gr_reads and gr_cl
piRNAs_hsa_fromPiC_f0 <- findOverlaps(align_69yo_allPrimary, clusters_69yo,
ignore.strand=FALSE)

mcols(align_69yo_allPrimary)$corr_piC_rankByReadsExplained <- 0

mcols(align_69yo_allPrimary)$corr_piC_rankByReadsExplained[queryHits(piRNAs_hsa_fromPiC_
f0)] <- mcols(clusters_69yo[subjectHits(piRNAs_hsa_fromPiC_f0)])$rank_readsExplained
length(align_69yo_allPrimary)

mcols(align_69yo_allPrimary)$piCgroup <- "NA"
mcols(align_69yo_allPrimary)$piCgroup[queryHits(piRNAs_hsa_fromPiC_f0)] <-
mcols(clusters_69yo[subjectHits(piRNAs_hsa_fromPiC_f0)])$piCgroup
length(align_69yo_allPrimary)
```

39486612

39486612

```
#check if all piRNA clusters present in align_69yo_allPrimary
setdiff(clusters_69yo$rank_readsExplained,
unique(align_69yo_allPrimary$corr_piC_rankByReadsExplained))
```

```
#add to readname rank_chr_startPos_strand_NH_piCgroup
seq <- align_69yo_allPrimary$seq
names(seq) <- paste0(names(align_69yo_allPrimary), "_rank",
align_69yo_allPrimary$corr_piC_rankByReadsExplained, "_",
seqnames(align_69yo_allPrimary), "_", start(align_69yo_allPrimary),
strand(align_69yo_allPrimary), "_NH", align_69yo_allPrimary$NH, "_piCgroup",
```

```
align_69yo_allPrimary$piCgroup)

writeXStringSet(seq,
file="../../../OneDrive/3_hsa_detection_prepach_adult/Code/STAROutput/oxSmallRNASeq/smal
lRNA_hsa_69y_A01_SRR8575350/smallRNA_hsa_69y_A01_SRR8575350_cleaned_trimmed_snoMiTRout_2
0to40nt_Aligned.PICBloadWithSeq.SeqsWithLocation.fasta")
```

Fasta just keeping nucleotides 2 to 20

```
align_69yo <-
readRDS("../../../OneDrive/3_hsa_detection_prepach_adult/Code/STAROutput/oxSmallRNASeq/s
mallRNA_hsa_69y_A01_SRR8575350/smallRNA_hsa_69y_A01_SRR8575350_cleaned_trimmed_snoMiTRou
t_20to40nt_Aligned.PICBloadWithSeq.rds")

align_69yo_allPrimary <- c(align_69yo$unique, align_69yo$multi.primary)
length(align_69yo_allPrimary)
```

```
39486612
```

```
#get cluster coordinates (ranked!) from Cell Reports publication (human 69 yo)
clusters_69yo <-
readRDS("../../../OneDrive/3_hsa_detection_prepach_adult/Code/STAROutput/oxSmallRNASeq/s
mallRNA_hsa_69y_A01_SRR8575350/clusters.ranked.smallRNA_hsa_69y_A01_SRR8575350_cleaned_t
rimmed_snoMiTRout_20to40nt_Aligned.sortedByCoord.ForceMapped.RDS")


clusters_69yo <- clusters_69yo[order(clusters_69yo$rank_readsExplained), ]
clusters_69yo$rank_readsExplained <- seq_along(clusters_69yo)
length(clusters_69yo)
```

```
13098
```

```
# load disentanglement_piCs_A01_69yo_SRR8575350_FM_90thPercentile_v2.RDS from
STAROutput/oxSmallRNASeq/smallRNA_hsa_69y_A01_SRR8575350/
piCs_A01_69yo_disentangled <-
readRDS("../../../OneDrive/3_hsa_detection_prepach_adult/Code/disentangledCluster/hsa_di
sentanglement_piCs_A01_69yo_SRR8575350_FM_90thPercentile_v2.RDS")

#add to readname rank_chr_startPos_strand_NH_piCgroup
#piCs_A01_69yo_disentangled is in three groups: unique69yo, commonShort, commonLong, add
this information (unique69yo="UN", commonSSC="CS", commonDynamic="CD") to another column
in clusters_69yo based on their shared column rank_readsExplained. If
rank_readsExplained is not listed in piCs_A01_69yo_disentangled, add "NA".
clusters_69yo$piCgroup <- "NA"
clusters_69yo$piCgroup[clusters_69yo$rank_readsExplained %in%
piCs_A01_69yo_disentangled$unique69yo$rank_readsExplained] <- "UN"
clusters_69yo$piCgroup[clusters_69yo$rank_readsExplained %in%
piCs_A01_69yo_disentangled$commonShort$rank_readsExplained] <- "CS"
clusters_69yo$piCgroup[clusters_69yo$rank_readsExplained %in%
piCs_A01_69yo_disentangled$commonLong$rank_readsExplained] <- "CD"
```

```
# Find overlaps between gr_reads and gr_cl
piRNAs_hsa_fromPiC_f0 <- findOverlaps(align_69yo_allPrimary, clusters_69yo,
ignore.strand=FALSE)

mcols(align_69yo_allPrimary)$corr_piC_rankByReadsExplained <- 0
```

```
mcols(align_69yo_allPrimary)$corr_piC_rankByReadsExplained[queryHits(piRNAs_hsa_fromPiC_
f0)] <- mcols(clusters_69yo[subjectHits(piRNAs_hsa_fromPiC_f0)])$rank_readsExplained
length(align_69yo_allPrimary)

mcols(align_69yo_allPrimary)$piCgroup <- "NA"
mcols(align_69yo_allPrimary)$piCgroup[queryHits(piRNAs_hsa_fromPiC_f0)] <-
mcols(clusters_69yo[subjectHits(piRNAs_hsa_fromPiC_f0)])$piCgroup
length(align_69yo_allPrimary)
```

```
39486612
39486612
```

```
#check if all piRNA clusters present in align_69yo_allPrimary
setdiff(clusters_69yo$rank_readsExplained,
unique(align_69yo_allPrimary$corr_piC_rankByReadsExplained))
```

```
#add to readname rank_chr_startPos_strand_NH_piCgroup
seq <- DNAStringSet(substring((align_69yo_allPrimary$seq), 2, 20))
names(seq) <- paste0(names(align_69yo_allPrimary), "_rank",
align_69yo_allPrimary$corr_piC_rankByReadsExplained, "_",
seqnames(align_69yo_allPrimary), "_", start(align_69yo_allPrimary),
strand(align_69yo_allPrimary), "_NH", align_69yo_allPrimary$NH, "_piCgroup",
align_69yo_allPrimary$piCgroup)

writeXStringSet(seq,
file="../../../OneDrive/3_hsa_detection_prepach_adult/Code/STAROutput/oxSmallRNASeq/smal
lRNA_hsa_69y_A01_SRR8575350/smallRNA_hsa_69y_A01_SRR8575350_cleaned_trimmed_snoMiTRout_2
0to40nt_Aligned.PICBloadWithSeq.SeqsWithLocation.2to20only.fasta")
```

Mapping to PCG-Exon-Transcriptome

Indexing Transcriptome (Only once)

Bash

```
#BASH, generateGenome
STAR --runMode genomeGenerate \
        --genomeDir
../../../OneDrive/General/hsa_referenceGenome/PCG_transcriptome/transcriptomeDir/ \
        --genomeSAindexNbases 6 \
        --genomeFastaFiles
../../../OneDrive/General/hsa_referenceGenome/PCG_transcriptome/hg38_PCGtranscriptome_co
llapsed_prioritizedCDS3UTR5UTR_allgenes.fasta \
        --limitGenomeGenerateRAM 34173092106\
        --runThreadN 23
```

```
Jan 04 08:43:54 ..... started STAR run
Jan 04 08:43:54 ... starting to generate Genome files
Jan 04 08:43:56 ... starting to sort Suffix Array. This may take a long time...
Jan 04 08:43:56 ... sorting Suffix Array chunks and saving them to disk...
Jan 04 08:44:59 ... loading chunks from disk, packing SA...
Jan 04 08:45:00 ... finished generating suffix array
Jan 04 08:45:00 ... generating Suffix Array index
Jan 04 08:45:00 ... completed Suffix Array index
Jan 04 08:45:00 ... writing Genome to disk ...
```

```
        Jan 04 08:45:00 ... writing Suffix Array to disk ...
        Jan 04 08:45:00 ... writing SAindex to disk
        Jan 04 08:45:00 ..... finished successfully
```

Mapping to Transcriptome

Bash

```
addFastaChange=""
addMappingChange="clip5pNbases1_Extend5pOfRead1_minMatch19"
```

```
input_fasta="../../../OneDrive/3_hsa_detection_prepach_adult/Code/STAROutput/oxSmallRNASeq/smallRNA_hsa_69y_A01_SRR8575350/smallRNA_hsa_69y_A01_SRR8575350_cleaned_trimmed_snoMiTRout_20to40nt_Aligned.PICBloadWithSeq.SeqsWithLocation${addFastaChange}.fasta"

#minimum Match of 19 nt, to restrict soft-clipping
STAR --runMode alignReads \
    --runThreadN 16 \
    --genomeDir
../../../OneDrive/General/hsa_referenceGenome/PCG_transcriptome/transcriptomeDir/ \
    --readFilesIn $input_fasta \
    --clip5pNbases 1 \
    --alignEndsType Extend5pOfRead1 \
    --outSAMattributes All \
    --outSAMtype BAM SortedByCoordinate \
    --limitBAMsortRAM 20000000000 \
    --alignIntronMax 1 \
    --alignSoftClipAtReferenceEnds No \
    --outFilterMismatchNmax 1 \
    --outFilterMatchNmin 19 \
    --winAnchorMultimapNmax 100 \
    --outFilterMultimapNmax 100 \
    --outReadsUnmapped Fastx \
    --outFileNamePrefix
pachTargetingGenes_PCG/hsa_69yo/PCG_smallRNA_hsa_69y_A01_SRR8575350_cleaned_trimmed_snoMiTRout_20to40nt_Aligned.PICBloadWithSeq.SeqsWithLocation.${addFastaChange}.PCGtranscriptome_${addMappingChange}_

echo "Mapped to PCG_EXON transcriptome"
```

```
suppressPackageStartupMessages({library(Rsamtools)})
#indexBam for every Bam in the folder pachTargetingGenes_PCG/hsa_69yo/ in R
for (bam in list.files("pachTargetingGenes_PCG/hsa_69yo", pattern="\\.bam$",
full.names=TRUE)) {
    indexBam(bam)
}
```

Which genes are targeted by pachytene piRNAs?

Prep - Load alignments and gene annotations

```
suppressPackageStartupMessages({
    library(Rsamtools)
    library(GenomicAlignments)
    library(GenomicRanges)
```

```
    library(PICB)
    library(BSgenome.Hsapiens.UCSC.hg38)
    library(tidyr)
    library(dplyr)
    library(ggplot2)
    library(ggrepel)
    library(patchwork)
    library(SVbyEye)
})
source("../scripts/plotCoverage.R")
```

```
# Load BAM file of piRNAs targeting protein coding genes
bamPCG_dir <-
"../data/bam/hsaToTranscriptome/PCG_smallRNA_hsa_69y_A01_SRR8575350_cleaned_trimmed_snoM
iTRout_20to40nt_Aligned.PICBloadWithSeq.SeqsWithLocation..PCGtranscriptome_clip5pNbases1
_Extend5pOfRead1_minMatch19_Aligned.sortedByCoord.out.bam"

bam <- BamFile(bamPCG_dir)
# Exclude both secondary alignments and supplementary alignments
fields <- scanBamWhat()
primary_flag <- scanBamFlag(isSecondary = FALSE, isSupplementary = FALSE)
param <- ScanBamParam(flag = primary_flag,
    what=c('qname','flag','rname','strand','pos','qwidth', 'cigar', 'seq'),
    tag=c('NH'))

ga_all_alignments <- readGAlignments(bam, param = param)
PCG_total_reads <- length(ga_all_alignments)
message("Number of reads mapped to PCG-transcriptome: ", PCG_total_reads)
ga_alignments <- ga_all_alignments[mcols(ga_all_alignments)$NH == 1] #UNIQUE ALIGNMENTS
ONLY!
PCG_unique_reads <- length(ga_alignments)
message("Number of reads mapped to PCG-transcriptome (unique alignments only): ",
PCG_unique_reads)

gr_alignments <- makeGRangesFromDataFrame(as.data.frame(ga_alignments),
keep.extra.columns=TRUE)

mcols(gr_alignments) <- mcols(gr_alignments)[ , !(colnames(mcols(gr_alignments)) %in%
c("njunc", "strandInfo", "rname", "strand.1", "pos", "qwidth.1", "cigar.1", "qual"))]
```

```
Number of reads mapped to PCG-transcriptome: 8995527

Number of reads mapped to PCG-transcriptome (unique alignments only): 7632221
```

```
geneAnnotation_PCG_EXON_dir <-
"../data/annotations/customTranscriptomes/hg38_PCGtranscriptome_collapsed_prioritizedCDS
3UTR5UTR_allgenes.gtf"

# geneAnnotation_PCG_EXON <- rtracklayer::import(geneAnnotation_PCG_EXON_dir)

# Import GTF file without rtracklayer (issues with installation)
read_gtf <- function(file_path) {
  # Read the GTF file - changed start/end to numeric first, then convert to integer
  gtf_data <- read.table(file_path, sep="\t", quote="",
                         col.names=c("seqname", "source", "feature", "start", "end",
                                     "score", "strand", "frame", "attribute"),
                         colClasses=c("character", "character", "character", "numeric",
```

```
"numeric",
                                    "character", "character", "character", "character"))

  # Convert coordinates to integer after reading
  gtf_data$start <- as.integer(gtf_data$start)
  gtf_data$end <- as.integer(gtf_data$end)

  # Function to extract attributes
  extract_attribute <- function(attr, key) {
    val <- sub(paste0(".*", key, "\\s+\"?([^;\"]+)\"?.*"), "\\1", attr)
    ifelse(val == attr, NA, val)
  }

  # Extract common attributes
  gtf_data$gene_id <- extract_attribute(gtf_data$attribute, "gene_id")
  gtf_data$transcript_id <- extract_attribute(gtf_data$attribute, "transcript_id")
  gtf_data$gene_name <- extract_attribute(gtf_data$attribute, "gene_name")

  return(gtf_data)
}

geneAnnotation_PCG <- read_gtf(geneAnnotation_PCG_EXON_dir)

geneAnnotation_PCG <- makeGRangesFromDataFrame(geneAnnotation_PCG,
                                               keep.extra.columns = TRUE,
                                               seqnames.field = "seqname",
                                               start.field = "start",
                                               end.field = "end",
                                               strand.field = "strand")

geneAnnotation_PCG_gene <- geneAnnotation_PCG[geneAnnotation_PCG$feature == "gene"]
```

```
# load genes gencode.v44.primary_assembly.annotation.gtf in hsa_referenceGenome
genes_dir <- "../data/annotations/gencode.v44.primary_assembly.annotation.gtf"
genes <- rtracklayer::import(genes_dir)
genes<-genes[genes$gene_name %in% unique(genes[genes$type %in% "CDS"]$gene_name)]
genes<-genes[genes$transcript_id %in% unique(genes[genes$type %in%
"CDS"]$transcript_id)] #since some transcripts do not contain all PCG annotation
features (CDS etc)
```

```
#piRNA cluster coordinates from Konstantinidou et al. (2024) in Cell Reports and ranked
by reads_explained (human 69 yo)
clusters_69yo <-
readRDS("../data/annotations/clusters.ranked.smallRNA_hsa_69y_A01_SRR8575350_cleaned_tri
mmed_snoMiTRout_20to40nt_Aligned.sortedByCoord.ForceMapped.RDS")

clusters_69yo <- clusters_69yo[order(clusters_69yo$rank_readsExplained), ]
clusters_69yo$rank_readsExplained <- seq_along(clusters_69yo)
clusters_69yo$gene_name <- paste0("piC-", clusters_69yo$rank_readsExplained)
length(clusters_69yo)
```

```
13098
```

```
gr_hg38 <- PICBload(
    BAMFILE =
"../data/bam/smallRNA_hsa_69y_A01_SRR8575350_cleaned_trimmed_snoMiTRout_20to40nt_Aligned
.sortedByCoord.out.bam",
```

```
    REFERENCE.GENOME = "BSgenome.Hsapiens.UCSC.hg38",
    GET.ORIGINAL.SEQUENCE = TRUE, VERBOSE = FALSE, WHAT = c("flag", "cigar"))

gr_hg38_prim <- c(gr_hg38$unique, gr_hg38$multi.primary)
genome_total_reads <- length(gr_hg38_prim)
message("Number of reads mapped to hg38 (primary alignments): ", genome_total_reads)
gr_hg38_prim <- keepStandardChromosomes(gr_hg38_prim)

gr_hg38 <- gr_hg38$unique
genome_unique_reads <- length(gr_hg38)
message("Number of reads mapped to hg38 (unique alignments only): ",
genome_unique_reads)
gr_hg38 <- keepStandardChromosomes(gr_hg38)
gr_hg38$qname <- names(gr_hg38)
```

```
Number of reads mapped to hg38 (primary alignments): 39486612

Number of reads mapped to hg38 (unique alignments only): 31897713
```

Genes targeted by piRNAs (Fig 4a)

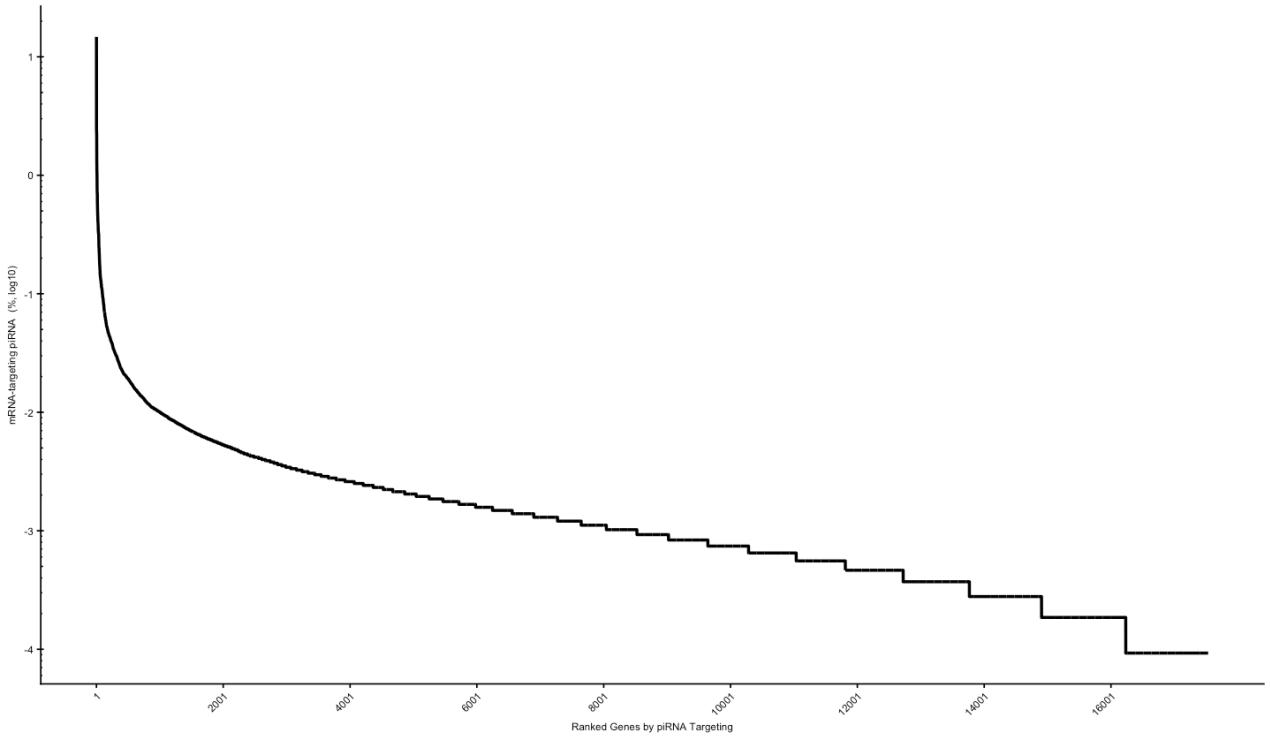
```
# Initialize result dataframe with all genes
topPiCtarget_df <- data.frame(
    geneName = geneAnnotation_PCG_gene$gene_name,
    topContribPiCrank = "0",
    topPercentage = 0,
    bypiCtargeting_percentage = 0,
    top_piC_percentage = 0,
    cisTargeting = FALSE,
    stringsAsFactors = FALSE
)

# Create function to get top contributing piC and percentage
get_top_piC_info <- function(overlaps) {
    if (length(overlaps) == 0) {
        return(c("0", 0, 0, 0))
    }
    # Extract rank information using vectorized operations
    ranks <- sub("_|_", "", sub("rank|_", "", regmatches(overlaps$qname,
regexpr("rank(.*?)_", overlaps$qname))))
    rank_table <- table(ranks)
    top_rank <- names(which.max(rank_table))
    top_percentage <- max(rank_table) / length(overlaps)
    # Get the percentage of targeting by piC
    bypiCtargeting_percentage <- sum(rank_table[names(rank_table) != "0"]) /
length(overlaps)

    if (top_rank == "0") {
        sorted_counts <- sort(rank_table, decreasing = TRUE)
        second_largest <- sorted_counts[2]
        top_piC_percentage <- second_largest / length(overlaps)
    } else {
        top_piC_percentage <- top_percentage
    }
    if (is.na(top_piC_percentage)) {
        top_piC_percentage <- 0
    }
```

```
    return(c(top_rank, top_percentage, top_piC_percentage, bypiCtargeting_percentage))
}

# Get all overlaps at once
all_overlaps <- findOverlaps(invertStrand(geneAnnotation_PCG_gene), gr_alignments)

# Split overlaps by gene
overlaps_by_gene <- split(gr_alignments[subjectHits(all_overlaps)],
                          queryHits(all_overlaps))

# Apply function to each gene's overlaps
results <- lapply(overlaps_by_gene, get_top_piC_info)

# Update only the rows that have overlaps
genes_with_overlaps <- as.numeric(names(overlaps_by_gene))
topPiCtarget_df$totalPiRNAcount <- countOverlaps(invertStrand(geneAnnotation_PCG_gene),
gr_alignments)
topPiCtarget_df$topContribPiCrank[genes_with_overlaps] <- sapply(results, `[`, 1)
topPiCtarget_df$topPercentage[genes_with_overlaps] <- as.numeric(sapply(results, `[`,
2))
topPiCtarget_df$bypiCtargeting_percentage[genes_with_overlaps] <-
as.numeric(sapply(results, `[`, 3))
topPiCtarget_df$top_piC_percentage[genes_with_overlaps] <- as.numeric(sapply(results,
`[`, 4))
rownames(topPiCtarget_df) <- topPiCtarget_df$geneName

# After updating topContribPiCrank and topPercentage, update cisTargeting
inverted_clusters <- invertStrand(clusters_69yo)
topContribPiCrank_numeric <- as.numeric(as.character(topPiCtarget_df$topContribPiCrank))

for (i in seq_len(nrow(topPiCtarget_df))) {
    geneNameI <- topPiCtarget_df$geneName[i]
    gene_overlaps <- subsetByOverlaps(
        inverted_clusters,
        genes[genes$gene_name == geneNameI])$rank_readsExplained
    topPiCtarget_df$cisTargeting[i] <- topContribPiCrank_numeric[i] %in% gene_overlaps
}
```

```
# Filter for non-cis targeted genes and order by totalPiRNAcount
targetingByPiRNAsSbO_sorted_total_tra <- topPiCtarget_df[topPiCtarget_df$cisTargeting ==
FALSE,]
message("Number of genes in antisense orientation to piRNA clusters and therefore
removed: ", nrow(topPiCtarget_df) - nrow(targetingByPiRNAsSbO_sorted_total_tra))
targetingByPiRNAsSbO_sorted_total_tra <-
targetingByPiRNAsSbO_sorted_total_tra[order(targetingByPiRNAsSbO_sorted_total_tra$totalP
iRNAcount, decreasing=TRUE),]

# Calculate percentage of targeting piRNAs for each gene
readsTargetingPCGs <- sum(targetingByPiRNAsSbO_sorted_total_tra$totalPiRNAcount)
targetingByPiRNAsSbO_sorted_total_tra$totalPiRNAperc <-
(targetingByPiRNAsSbO_sorted_total_tra$totalPiRNAcount/readsTargetingPCGs)*100
targetingByPiRNAsSbO_sorted_total_tra$targetedRank <-
1:nrow(targetingByPiRNAsSbO_sorted_total_tra)
genes_targetingThreshold <- 50
```

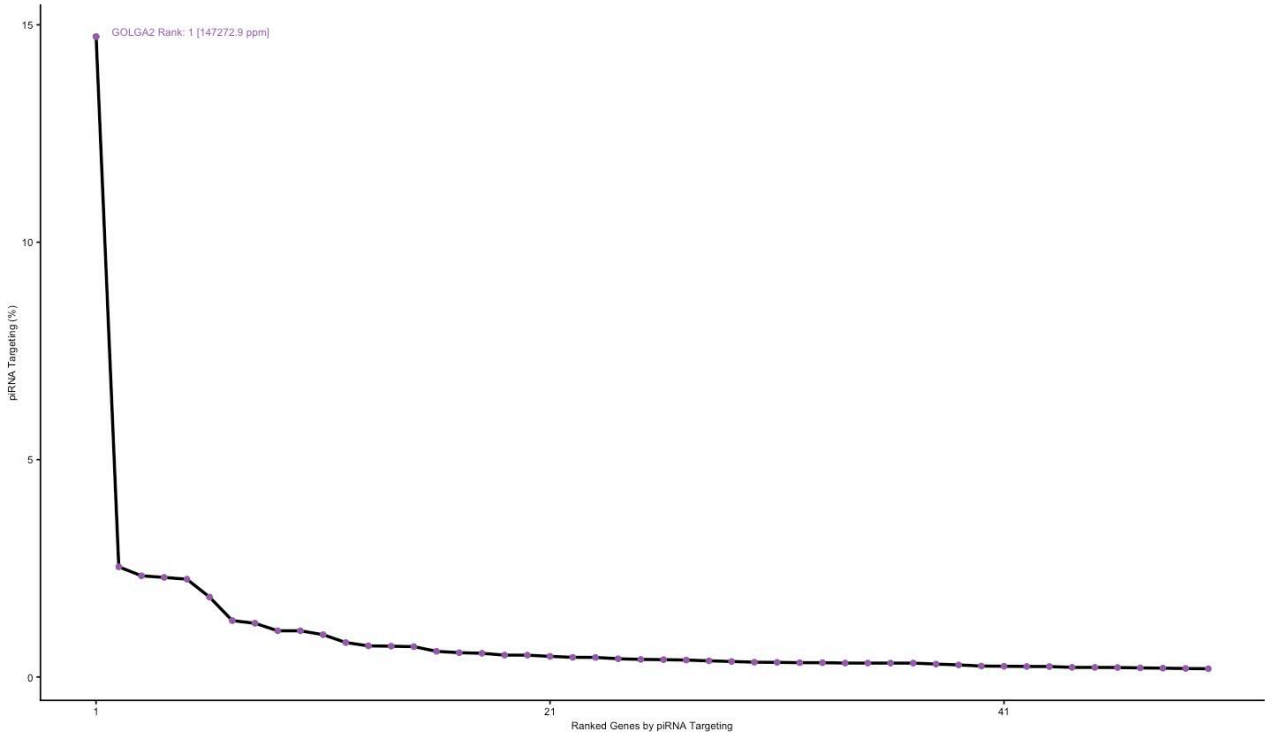
```
Number of genes in antisense orientation to piRNA clusters and therefore remove
d: 697
```

```
options(repr.plot.width=12, repr.plot.height=7)
# Create plot with highlighted genes and corrected ranks
plot_4a_full <-
ggplot(targetingByPiRNAsSbO_sorted_total_tra[targetingByPiRNAsSbO_sorted_total_tra$total
PiRNAcount != 0,],
    aes(x = targetedRank, y = log10(totalPiRNAperc), group = 1)) +
    geom_vline(xintercept = 0:genes_targetingThreshold, color = "#ffffff", alpha = 0.3)
+
    scale_x_continuous(breaks = seq(1,
nrow(targetingByPiRNAsSbO_sorted_total_tra[targetingByPiRNAsSbO_sorted_total_tra$totalPi
RNAcount != 0,]), by = 2000)) +
    geom_line(linewidth = 1) +
    theme_classic() +
    annotation_logticks(base = 10, sides = "l", short = unit(0.02, "cm"), mid =
unit(0.04, "cm"), long = unit(0.06, "cm")) +
    theme(
        axis.text.x = element_text(angle = 45, hjust = 1, size = 7),
        axis.text.y = element_text(size = 7),
        axis.title.x = element_text(size = 7),
        axis.title.y = element_text(size = 7)
    ) +
    labs(
        x = "Ranked Genes by piRNA Targeting",
        y = "mRNA-targeting piRNA  (%, log10)"
    )

plot_4a_full
```




```
options(repr.plot.width=12, repr.plot.height=7)

# Get the data for specified genes
genes_targetingThreshold <- 50
highlight_genes <- c('GOLGA2')
gene_data <- targetingByPiRNAsSbO_sorted_total_tra[
```

```
    targetingByPiRNAsSbO_sorted_total_tra$geneName %in% highlight_genes,]

# Create plot with highlighted genes and corrected ranks
plot_4aInset <-
ggplot(targetingByPiRNAsSbO_sorted_total_tra[1:genes_targetingThreshold,],
    aes(x = targetedRank, y = totalPiRNAperc, group = 1)) +
    geom_line(linewidth = 1, color = "black") +
    geom_point(color = "#9662A9", size = 1.5) +
    scale_x_continuous(breaks = seq(1, nrow(targetingByPiRNAsSbO_sorted_total_tra), by =
20)) +
    # Add highlighted points with different color/size to make them stand out
    geom_point(data = gene_data, color = "#9662A9", size = 1.5) +
    # Add labels for highlighted genes
    geom_text(data = gene_data,
              aes(label = sapply(geneName, function(g) {
                  count <-
targetingByPiRNAsSbO_sorted_total_tra$totalPiRNAcount[targetingByPiRNAsSbO_sorted_total_
tra$geneName == g]
                  ppm <- round((count / readsTargetingPCGs) * 1e6, 1)
                  paste0(g, " Rank: ",
                         match(g, targetingByPiRNAsSbO_sorted_total_tra$geneName),
                         " [", ppm, " ppm]")
              })),
              color = "#9662A9",
              vjust = -0.2,  # Single value for single gene
              hjust = -0.1,  # Single value for single gene
              size = 2.5) +
    theme_classic() +
    theme(
        axis.text.x = element_text(size = 7),
        axis.text.y = element_text(size = 7),
        axis.title.x = element_text(size = 7),
        axis.title.y = element_text(size = 7)
    ) +
    labs(
        x = "Ranked Genes by piRNA Targeting",
        y = "piRNA Targeting (%)"
    )

plot_4aInset
```



Scatter plot of mRNA target length and the log10-transformed percentage of targeting piRNAs (Fig 4b)

```
#load pseudogenes gencode.v47.2wayconspseudos.gtf in hsa_referenceGenome
pseudogenes_dir <- "../data/annotations/gencode.v47.2wayconspseudos.gtf"
pseudogenes <- rtracklayer::import(pseudogenes_dir)

# pseudogenes don't contain regular gene_name of parent_id (just Ensembl format)
cleaned_gene_ids_genes <- sub("\\..*", "", mcols(genes)$gene_id)
gene_id_to_name_map <- setNames(mcols(genes)$gene_name, cleaned_gene_ids_genes)

# Add the 'geneName' column to 'pseudogenes'
mcols(pseudogenes)$geneName <- ifelse(
    mcols(pseudogenes)$parent_id %in% names(gene_id_to_name_map),
    gene_id_to_name_map[mcols(pseudogenes)$parent_id],
    mcols(pseudogenes)$parent_id
)
```

```
# prefilter piRNA clusters that are generally targeting PCG genes
maxPiC <-
max(as.integer(targetingByPiRNAsSbO_sorted_total_tra[1:250,]$topContribPiCrank))
message("Only considering top ", maxPiC, " piRNA clusters since top 250 targeted genes
only are targeted by that max rank.")
clusters_69yo_pachSubset <- clusters_69yo[clusters_69yo$rank_readsExplained <= maxPiC]
pseudogenesRGOvrlp <- subsetByOverlaps(pseudogenes,
invertStrand(clusters_69yo_pachSubset))
```

```
Only considering top 8406 piRNA clusters since top 250 targeted genes only are t
argeted by that max rank.
```

```
# Find overlaps
overlaps <- findOverlaps(pseudogenesRGOvrlp, invertStrand(clusters_69yo_pachSubset))
```

```
# Extract ranks based on overlaps
ranks <- rep(NA, length(pseudogenesRGOvrlp))
ranks[queryHits(overlaps)] <-
clusters_69yo_pachSubset$rank_readsExplained[subjectHits(overlaps)]

# Add rank to pseudogenesRGOvrlp
pseudogenesRGOvrlp$rank_readsExplained <- ranks
```

```
# Get unique pairs
unique_pairs <- unique(data.frame(
    parentName = mcols(pseudogenesRGOvrlp)$geneName,
    rank = mcols(pseudogenesRGOvrlp)$rank_readsExplained
))
# Sort by cluster_rank if desired
unique_pairs <- unique_pairs[order(unique_pairs$rank), ]

unique_pairs$geneTop250targeted <- unique_pairs$parentName %in% topPiCtarget_df$geneName
```

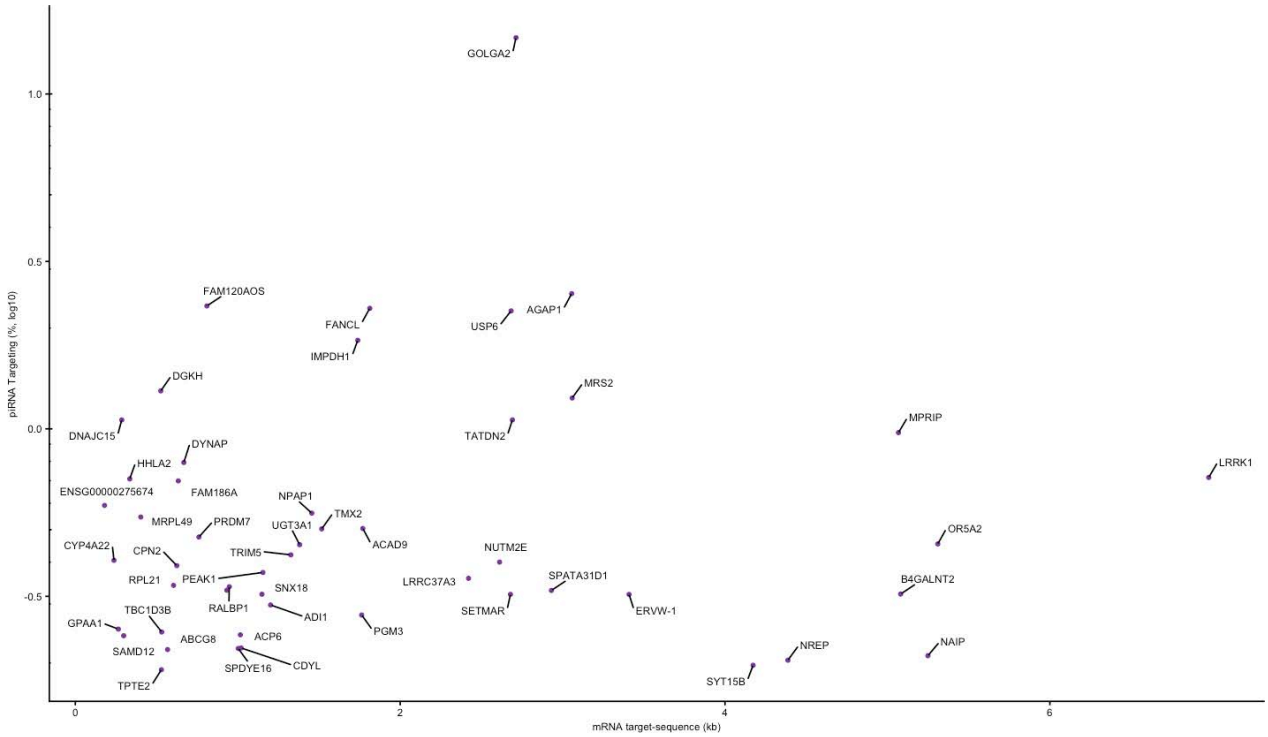
```
# Initialize PGasToTopPiC, check if gene is targeted by a piC that contains its
Pseudogene
targetingByPiRNAsSbO_sorted_total_tra$PGasToTopPiC <- "NO"
targetingByPiRNAsSbO_sorted_total_tra$PGasToTopPiC[targetingByPiRNAsSbO_sorted_total_tra
$topContribPiCrank == 0] <- "N.A."

for (gene in targetingByPiRNAsSbO_sorted_total_tra$geneName) {
    if (gene %in% unique(unique_pairs$parentName)) {
        assRank <- na.omit(unique_pairs[unique_pairs$parentName == gene, "rank"])
        for (rank in assRank) {
            if
(targetingByPiRNAsSbO_sorted_total_tra[targetingByPiRNAsSbO_sorted_total_tra$geneName ==
gene,]$topContribPiCrank == rank) {

targetingByPiRNAsSbO_sorted_total_tra[targetingByPiRNAsSbO_sorted_total_tra$geneName ==
gene,]$PGasToTopPiC <- "YES"
            }
        }
    }
}

#mark also genes that have unannotated Pseudogenes (GOLGA2 is annotated for piC-ranked 1
in UCSC), by changing the BarColorPseudogene to purple
targetingByPiRNAsSbO_sorted_total_tra$PGasToTopPiC[targetingByPiRNAsSbO_sorted_total_tra
$geneName == "GOLGA2"] <- "YES"
```

```
#get total coverage of targeting of gene

gr_alignments_red <- reduce(gr_alignments)

# Calculate overlap widths for matches
overlapsPiRNAsRedGenes <- findOverlaps(gr_alignments_red,
invertStrand(geneAnnotation_PCG_gene))
overlap_widths <-
tapply(width(pintersect(gr_alignments_red[queryHits(overlapsPiRNAsRedGenes)],

invertStrand(geneAnnotation_PCG_gene[subjectHits(overlapsPiRNAsRedGenes)])))),
```

```
                        subjectHits(overlapsPiRNAsRedGenes), sum)

# Create a vector of length equal to number of genes in geneAnnotation_PCG_gene
full_overlap_widths <- numeric(length(geneAnnotation_PCG_gene))

# Fill in the actual overlap values where they exist
full_overlap_widths[as.numeric(names(overlap_widths))] <- overlap_widths

# Create dataframe with all genes and their coverage
all_genes <- geneAnnotation_PCG_gene$gene_name
percentageCoverage_df <- data.frame(
    geneName = all_genes,
    targetingCoverage_bp = full_overlap_widths
)

# Merge with targeting dataframe
targetingByPiRNAsSbO_sorted_total_tra <- merge(
    targetingByPiRNAsSbO_sorted_total_tra,
    percentageCoverage_df,
    by = "geneName"
)

targetingByPiRNAsSbO_sorted_total_tra <-
targetingByPiRNAsSbO_sorted_total_tra[order(targetingByPiRNAsSbO_sorted_total_tra$totalP
iRNAcount, decreasing=TRUE),]
```

```
plot_targetingCoverage <-
ggplot(targetingByPiRNAsSbO_sorted_total_tra[1:genes_targetingThreshold,], aes(x =
targetingCoverage_bp/1000, y = log10(totalPiRNAperc))) +
  geom_point(aes(color = "#7f3f98"), alpha = 1, size = 1) +
  geom_text_repel(aes(label = geneName),
                  size = 2.5,
                  box.padding = 0.5,
                  max.overlaps = 10) +
  annotation_logticks(base = 10, sides = "l", short = unit(0.02, "cm"), mid = unit(0.04,
"cm"), long = unit(0.06, "cm")) +
  scale_color_identity(guide = "legend",
                       breaks = c("#c353ff","#d3d3d3",  "#767676"),
                       labels = c("PG-as piCs-derived piRNAs",
                                  "non-piC-derived piRNAs",
                                  "piC-derived piRNAs")) +
  theme_classic() +
  labs(
    x = "mRNA target-sequence (kb)",
    y = "piRNA Targeting (%, log10)"
  ) +
  theme(
    plot.title = element_text(size = 7, face = "bold"),
    axis.title = element_text(size = 7),
    axis.text = element_text(size = 7),
    legend.position = "none"
  )

plot_targetingCoverage
```

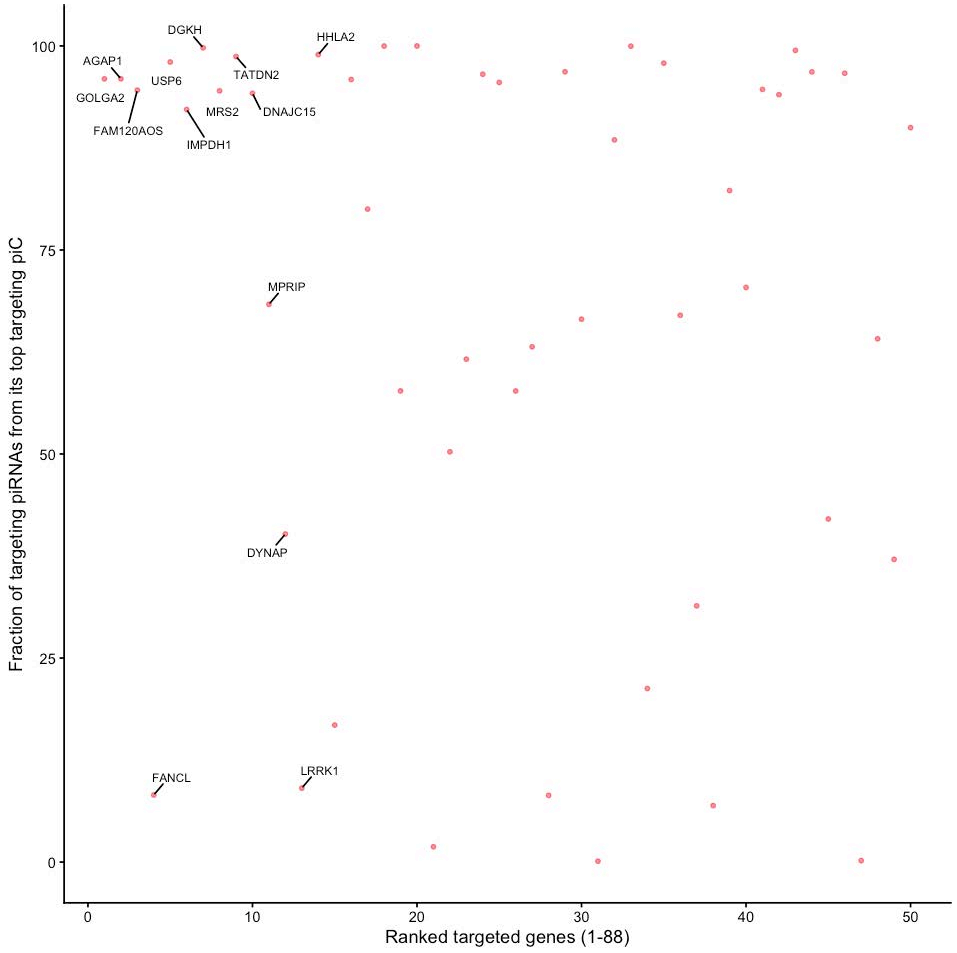
```
Warning message:
"ggrepel: 2 unlabeled data points (too many overlaps). Consider increasing max.o
verlaps"
```



piRNA Cluster origins of targeting piRNAs (Fig 4b-Inset, 4c, Extended Data Fig. 4d)

Fraction of piRNAs derived from their most targeting piRNA cluster

```
options(repr.plot.width=8, repr.plot.height=8)

# only label dots that rank 15 or higher in targeting
temp_targetingByPiRNAsSbO_sorted_total_tra <- targetingByPiRNAsSbO_sorted_total_tra %>%
  mutate(label = if_else(targetedRank < 15, geneName, NA_character_))

# plot fraction of targeting piRNAs from its top targeting piRNA cluster
plot_scTopPer <-
ggplot(temp_targetingByPiRNAsSbO_sorted_total_tra[1:genes_targetingThreshold,], aes(x =
targetedRank, y = top_piC_percentage*100)) +
    geom_point(color = "#ed1c24", alpha = 0.5, size = 1) +
    geom_text_repel(aes(label = label),
                    size = 2.5,
                    box.padding = 0.5,
                    max.overlaps = 10,
                    na.rm = TRUE) +
    scale_color_identity() +
    theme_classic() +
    labs(y = "Fraction of targeting piRNAs from its top targeting piC", x = "Ranked
targeted genes (1-88)") +
    ylim(0, 100)

plot_scTopPer
```



Fraction of gene-targeting piRNAs that originate from piRNA clusters that are not directly overlapping with the specific gene (in trans)

```
# put alignments to PCG transcriptome into context with the genes they target
# match read names, which include (among other things) information about the piC they
came from (rank) and their original read name (when mapped to mm10)
PCG_as_name_df <- as.data.frame(findOverlaps(gr_alignments,
invertStrand(geneAnnotation_PCG_gene)))
PCG_as_name_df$gene_name <-
geneAnnotation_PCG_gene[as.numeric(PCG_as_name_df$subjectHits), ]$gene_id
PCG_as_name_df$read_name <- sub("_.*", "",
gr_alignments[as.numeric(PCG_as_name_df$queryHits),]$qname)
PCG_as_name_df$read_nameWInfo <-
gr_alignments[as.numeric(PCG_as_name_df$queryHits),]$qname
PCG_as_name_df$rankOrigin <- sub("_|_", "", sub("rank|_", "",
regmatches(gr_alignments[as.numeric(PCG_as_name_df$queryHits),]$qname,
regexpr("rank(.*?)_", gr_alignments[as.numeric(PCG_as_name_df$queryHits),]$qname))))

# Precompute read–gene pairs from a single overlap against invertStrand(genes)
inv_genes <- invertStrand(genes)
hits <- findOverlaps(gr_hg38_prim, inv_genes)

# get read name - gene pairs that are antisense to eachother in mm10
hg38_as_pair <- unique(paste0(names(gr_hg38_prim)[as.integer(queryHits(hits))], "\r",
                              inv_genes$gene_name[as.integer(subjectHits(hits))]))

# Mark cis if (read_name, gene_name) observed in the precomputed pairs
```

```
PCG_as_pair <- paste0(PCG_as_name_df$read_name, "\r", PCG_as_name_df$gene_name)
PCG_as_name_df$cis_piRNA <- PCG_as_pair %in% hg38_as_pair
```

```
# all genes
# targeting in trans
nrow(PCG_as_name_df[!PCG_as_name_df$cis_piRNA,])/nrow(PCG_as_name_df)
# targeting in trans and from piRNA cluster
nrow(PCG_as_name_df[(!PCG_as_name_df$cis_piRNA) & (PCG_as_name_df$rankOrigin !=
0),])/nrow(PCG_as_name_df[!PCG_as_name_df$cis_piRNA,])
```

0.596092665642742

0.646610386344729

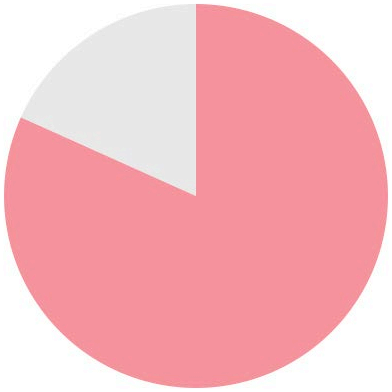
```
# top targeted genes (50)
PCG_as_name_df_topTargetedGenes <- PCG_as_name_df[PCG_as_name_df$gene_name %in%
targetingByPiRNAsSbO_sorted_total_tra$geneName[1:genes_targetingThreshold], ]

# targeting in trans
nrow(PCG_as_name_df_topTargetedGenes[!PCG_as_name_df_topTargetedGenes$cis_piRNA,])/nrow(
PCG_as_name_df_topTargetedGenes)
# targeting in trans and from piRNA cluster
nrow(PCG_as_name_df_topTargetedGenes[(!PCG_as_name_df_topTargetedGenes$cis_piRNA) &
(PCG_as_name_df_topTargetedGenes$rankOrigin !=
0),])/nrow(PCG_as_name_df_topTargetedGenes[!PCG_as_name_df_topTargetedGenes$cis_piRNA,])
```

0.954526083527828

0.817176798012188

```
# for top-targeted genes, fraction targeted by piRNAs from piRNA cluster (given that
they are trans targeting)
PCG_as_name_df_topTargetedGenes <- PCG_as_name_df[PCG_as_name_df$gene_name %in%
targetingByPiRNAsSbO_sorted_total_tra$geneName[1:genes_targetingThreshold], ]
transTargetingPiCpiRNAs <-
nrow(PCG_as_name_df_topTargetedGenes[(!PCG_as_name_df_topTargetedGenes$cis_piRNA) &
(PCG_as_name_df_topTargetedGenes$rankOrigin !=
0),])/nrow(PCG_as_name_df_topTargetedGenes[!PCG_as_name_df_topTargetedGenes$cis_piRNA,])
transTargetingPiCpiRNAs
# Create a data frame with this value
data <- data.frame(
  category = c("transTargetingPiCpiRNAs", "Other"),
  value = c(transTargetingPiCpiRNAs, 1-transTargetingPiCpiRNAs)
)

# Create the pie chart
options(repr.plot.width=4, repr.plot.height=4)
plot_byPiCTarg <- ggplot(data, aes(x = "", y = value, fill = category)) +
  geom_bar(stat = "identity", width = 1, alpha=0.5) +
  coord_polar(theta = "y") +
  scale_fill_manual(values = c("lightgrey", "#ed1c24")) +
  theme_void() +
  theme(legend.position = "none")

plot_byPiCTarg
```

0.817176798012188



Contribution of individual piRNA clusters to targeting (by their top, second, and third most contributing piC, other piCs and non-piC regions)

```
# filter custom PCG transcriptome coordinates of top targeted genes
subset_tra_gr <- geneAnnotation_PCG_gene[geneAnnotation_PCG_gene$gene_name %in%
targetingByPiRNAsSbO_sorted_total_tra[1:genes_targetingThreshold,]$geneName]
# add totalPiRNAcounts and targetedRank to GRange object
mcols(subset_tra_gr)$totalPiRNAcount <-
targetingByPiRNAsSbO_sorted_total_tra$totalPiRNAcount[match(mcols(subset_tra_gr)$gene_na
me, targetingByPiRNAsSbO_sorted_total_tra$geneName)]
mcols(subset_tra_gr)$targetedRank <-
targetingByPiRNAsSbO_sorted_total_tra$targetedRank[match(mcols(subset_tra_gr)$gene_name,
targetingByPiRNAsSbO_sorted_total_tra$geneName)]
```

```
# get table with piC-rank targeting contributions per gene
rank_piC_info <- function(overlaps) {
    if (length(overlaps) == 0) {
        return(c("0", 0, 0))
    }
    # Extract rank information using vectorized operations
    ranks <- sub("_|_", "", sub("rank|_", "", regmatches(overlaps$qname,
regexpr("rank(.*?)_", overlaps$qname))))
    rank_table <- table(ranks)

    return(rank_table)
}

# Get all targeting piRNAs in relation to the gene they target
overlap_hits <- findOverlaps(invertStrand(subset_tra_gr), gr_alignments)

# Split targeting piRNAs by gene
alignments_by_gene <- split(gr_alignments[subjectHits(overlap_hits)],
                            queryHits(overlap_hits))

# For each gene run rank_piC_info for table with piC-rank targeting contributions
piC_rank_summaries <- lapply(alignments_by_gene, rank_piC_info)
```

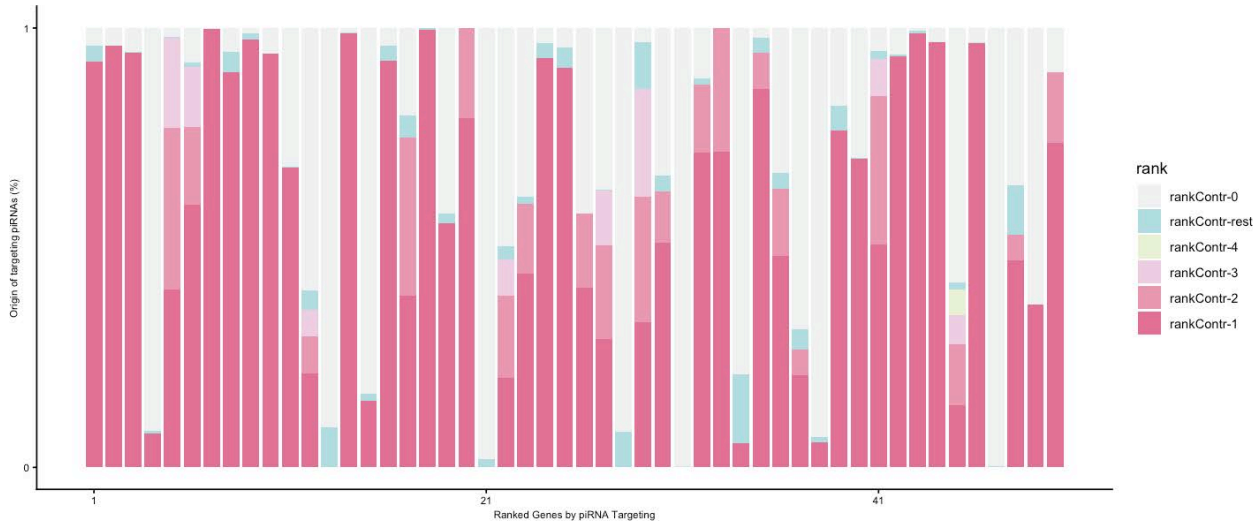
```
# Retrieve top contributing piC and percentage
# Separate piRNAs not from piCs and those from piCs that contribute < 5%
process_rank_contributions <- function(rank_table, total_count) {
    # Convert gene's piC_rank_summaries table to named vector and calc fractions
```

```
    contributions <- as.vector(rank_table) / total_count
    names(contributions) <- names(rank_table)

    # Separate category 0 (if it exists)
    cat_0 <- if("0" %in% names(contributions)) contributions["0"] else 0
    other_contributions <- contributions[names(contributions) != "0"]

    # Sort other contributions in descending order
    sorted_contributions <- sort(other_contributions, decreasing = TRUE)

    # Identify contributions >= 5%
    major_contributions <- sorted_contributions[sorted_contributions >= 0.05]
    minor_contributions <- sorted_contributions[sorted_contributions < 0.05]

    # Create result vector
    result <- c()
    result["rankContr-0"] <- cat_0

    # Add major contributions
    for(i in seq_along(major_contributions)) {
        result[paste0("rankContr-", i)] <- major_contributions[i]
    }

    # Sum minor contributions if any exist
    if(length(minor_contributions) > 0) {
        result["rankContr-rest"] <- sum(minor_contributions)
    }

    return(result)
}
```

```
# Apply to each gene and create new columns
contribution_results <- lapply(seq_along(alignments_by_gene), function(i) {
    rank_table <- piC_rank_summaries[[i]]
    total_count <- subset_tra_gr$totalPiRNAcount[i]
    process_rank_contributions(rank_table, total_count)
})
```

```
# Find all unique column names across all results
all_columns <- unique(unlist(lapply(contribution_results, names)))

# Ensure each result has all columns, filling missing ones with 0
contribution_results_normalized <- lapply(contribution_results, function(x) {
    missing_cols <- setdiff(all_columns, names(x))
    if(length(missing_cols) > 0) {
        x[missing_cols] <- 0
    }
    return(x[all_columns])
})
# convert to one data frame
contribution_df <- do.call(rbind, contribution_results_normalized)
colnames(contribution_df) <- all_columns

# Add contribution_df to the subset_tra_gr in df format
subset_tra <- cbind(as.data.frame(subset_tra_gr), contribution_df)
```

```r
# Reshape to long format
data_long <- subset_tra %>%
  mutate(gene = rownames(subset_tra)) %>%
  gather(key = "rank", value = "value",
         starts_with("rankContr")) %>%
  mutate(rank = factor(rank,
                       levels = c("rankContr-0", "rankContr-rest", "rankContr-4",
"rankContr-3", "rankContr-2", "rankContr-1")))

# Create the stacked column chart
options(repr.plot.width=12, repr.plot.height=5)
plot_piCcontr <- ggplot(data_long, aes(x = targetedRank, y = value, fill = rank)) +
  geom_col(width = 0.85) +
  scale_y_continuous(breaks = c(0, 1), labels = c("0", "1")) +
    scale_x_continuous(breaks = seq(1, nrow(subset_tra), by = 20)) +
  scale_fill_manual(values = c("#F1F2F2", "#b3dee2", "#eaf2d7", "#efcfe3",
                               "#ea9ab2", "#e27396")) +
  theme_classic() +
  theme(
    axis.text = element_text(size = 7),
    axis.title = element_text(size = 7),
  ) +
  labs(x = "Ranked Genes by piRNA Targeting",
       y = "Origin of targeting piRNAs (%)")


plot_piCcontr
```



Targeting of gene features (Extended Data Fig. 4a)

```r
geneAnnotation_PCG_gene_subset <-
geneAnnotation_PCG_gene[geneAnnotation_PCG_gene$gene_name %in%
targetingByPiRNAsSbO_sorted_total_tra$geneName[1:250]]
selected_genes <- geneAnnotation_PCG_gene_subset$gene_name
```

```r
unique(geneAnnotation_PCG$feature)
```

'gene' · '5UTR' · 'CDS' · '3UTR'

```
# subset geneAnnotation_PCG by genes (top targeted)
# and features (removing 'gene' annotation which includes all collapsed exons, not
seperated by features)
feature_list <- c("5UTR", "CDS", "3UTR")
gene_features <- geneAnnotation_PCG[geneAnnotation_PCG$gene_name %in% selected_genes &
geneAnnotation_PCG$feature %in% feature_list]
```

```
#pre-select alignments to only include alignments targeting selected genes
gr_alignments_main <- subsetByOverlaps(gr_alignments,
invertStrand(geneAnnotation_PCG_gene_subset))
mcols(gr_alignments_main) <- NULL
```

```
# make GRanges that have the starting position for each targeting piRNA
# For positive strand, make end = start
end(gr_alignments_main)[as.character(strand(gr_alignments_main)) == "+"] <-
start(gr_alignments_main)[as.character(strand(gr_alignments_main)) == "+"]

# For negative strand, make start = end
start(gr_alignments_main)[as.character(strand(gr_alignments_main)) == "-"] <-
end(gr_alignments_main)[as.character(strand(gr_alignments_main)) == "-"]
```

```
all_tiles <- NULL
# iterate through each gene
for (gene in unique(gene_features$gene_id)) {
    # get all piRNAs targeting that gene
    piRNA_startsAsToGene <- subsetByOverlaps(gr_alignments_main,
invertStrand(gene_features[gene_features$gene_id == gene]))

    # get gene coordinates in custom PCG transcriptome
    gene_ranges <- gene_features[gene_features$gene_id == gene]

    # iterate through each feature
    for (feature in feature_list) {

        # Subset to gene's feature
        temp_gr <- gene_ranges[gene_ranges$feature == feature]
        if (length(temp_gr) == 0) {
            # skip if gene does not have 5'UTR or 3'UTR
            next
        }

        # Merge overlapping or adjacent ranges
        merged_gr <- reduce(temp_gr)
        total_length <- sum(width(merged_gr))

        if (total_length < 20) {
            cat("   Feature ", feature, " for gene ", gene, " too short (<20 nt total).
Skipping.\n")
            next
        }

        # Figure out which direction to tile
        gene_strand <- unique(as.character(strand(temp_gr)))
        merged_gr <- sort(merged_gr)
```

```
# Determine exact tile sizes so that each tile is ~5%
base_tile_size <- floor(total_length / 20)
leftover <- total_length %% 20
tile_sizes <- rep(base_tile_size, 20)

# Distribute the remainder (leftover) among the first tiles
if (leftover > 0) {
    if (gene_strand == "+") {
        tile_sizes[seq_len(leftover)] <- tile_sizes[seq_len(leftover)] + 1
    } else {
        tile_sizes[21-seq_len(leftover)] <- tile_sizes[21-seq_len(leftover)] + 1
    }
}

# Build the 20 tiles by walking through merged_gr
tile_list <- vector("list", 20)
current_tile_index <- 1
target_tile_len <- tile_sizes[current_tile_index]
cum_len_in_tile    <- 0
current_ranges     <- IRanges()

# Helper to finalize a tile and reset
finalize_tile <- function() {
    tile_list[[current_tile_index]] <<- GRanges(
        seqnames = seqnames(merged_gr)[1],
        ranges   = current_ranges,
        strand   = gene_strand,
        gene_id  = gene,
        feature  = feature,
        tile_index = current_tile_index
    )

    current_tile_index <<- current_tile_index + 1
    if (current_tile_index <= 20) {
        target_tile_len <<- tile_sizes[current_tile_index]
    }
    cum_len_in_tile <<- 0
    current_ranges <<- IRanges()
}

for (seg in seq_along(merged_gr)) {
    seg_start <- start(merged_gr[seg])
    seg_end   <- end(merged_gr[seg])
    seg_width <- width(merged_gr[seg])

    bases_used_in_seg <- 0

    # Iterate base by base in principle, but slice big chunks if possible
    while (bases_used_in_seg < seg_width && current_tile_index <= 20) {

        # Still need 'remaining_in_tile' bases to complete the current tile
        needed_for_tile <- target_tile_len - cum_len_in_tile
        # The maximum we can take from the current segment is what's left in it
        left_in_segment <- seg_width - bases_used_in_seg
        # The actual chunk we'll consume from this segment
        chunk_size <- min(needed_for_tile, left_in_segment)

        if (chunk_size == 0) {
            # tile is exactly filled
            finalize_tile()
```

```
                    if (current_tile_index > 20) break
                        next
                }


                chunk_start <- seg_start + bases_used_in_seg
                chunk_end   <- chunk_start + chunk_size - 1


                # Add IRanges chunk
                current_ranges <- c(
                current_ranges,
                IRanges(start = chunk_start, end = chunk_end)
                )

                # Update counters
                bases_used_in_seg <- bases_used_in_seg + chunk_size
                cum_len_in_tile    <- cum_len_in_tile + chunk_size

                # If tile is filled, finalize
                if (cum_len_in_tile == target_tile_len) {
                    finalize_tile()
                    if (current_tile_index > 20) break
                }
            }
            if (current_tile_index > 20) break
        }

        # If something left in the last tile
        if (current_tile_index <= 20 && cum_len_in_tile > 0) {
            finalize_tile()
        }

        final_tiles <- do.call(c, tile_list)  # a GRanges of length 20

        #handle minus strand by just reversing the column tile_index (20 to 1, 19 to 2,
etc)
        if (gene_strand == "-") {
            final_tiles$tile_index <- 20 - final_tiles$tile_index + 1
        }

        # Count overlaps for each tile and add these counts to metacolumn of final_tiles
        overlap_counts <- countOverlaps(final_tiles, invertStrand(gr_alignments_main))
        mcols(final_tiles)$read_counts <- overlap_counts

        #combine to all_tiles
        all_tiles <- c(all_tiles, final_tiles)

  }
}
all_tiles <- do.call(c, all_tiles)
```

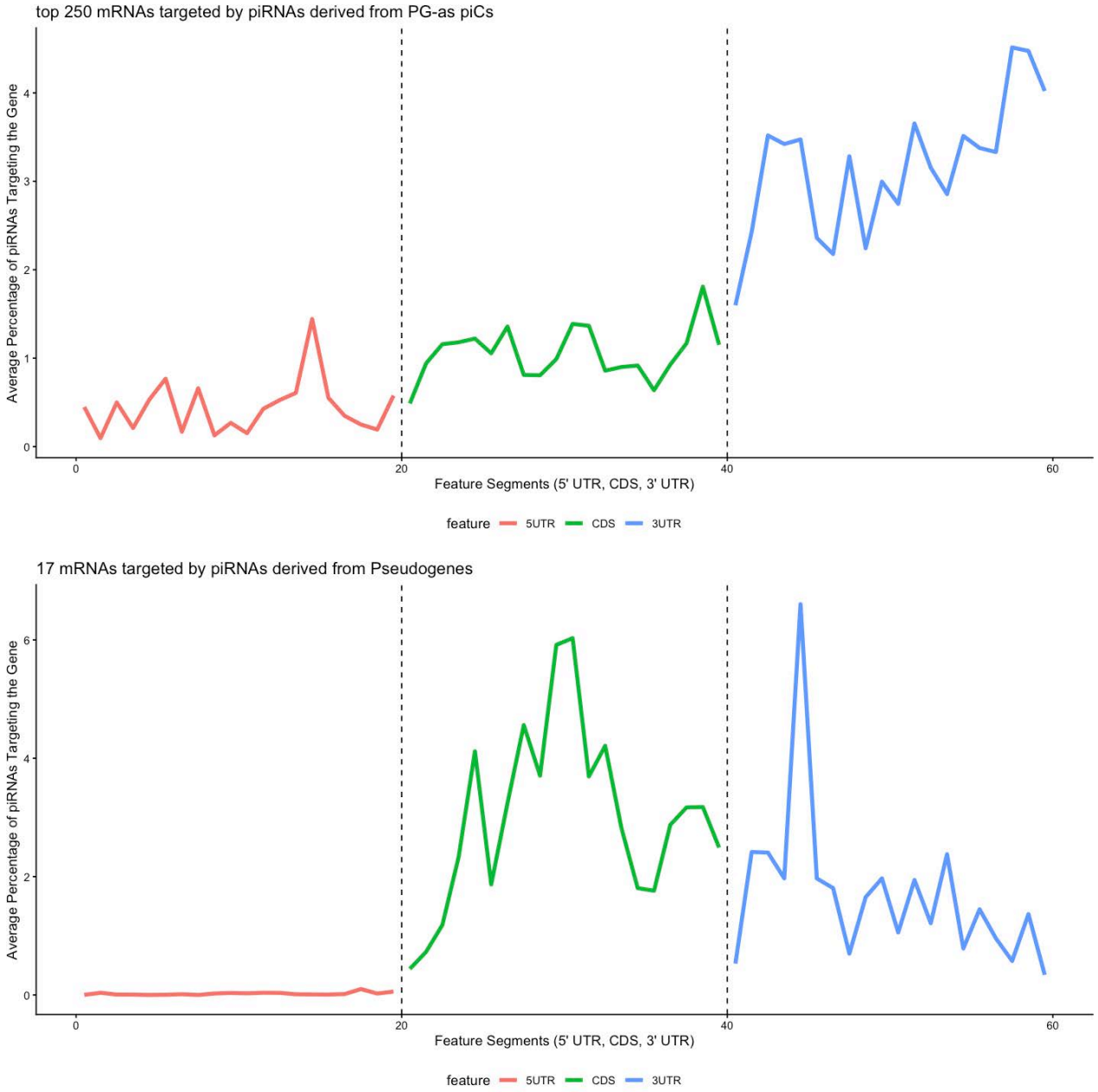
```
Feature  5UTR  for gene  RPAP2  too short (<20 nt total). Skipping.
Feature  5UTR  for gene  KIAA1143  too short (<20 nt total). Skipping.
Feature  3UTR  for gene  ENSG00000281039  too short (<20 nt total). Skipping.
```

```
# Calc percentages of targeting piRNAs for each gene
all_tiles_df <- as.data.frame(all_tiles) %>%
  group_by(gene_id) %>%
  mutate(total_counts = sum(read_counts),
```

```
         percentage = (read_counts / total_counts) * 100)

all_tiles_df$feature <- factor(all_tiles_df$feature, levels = c("5UTR", "CDS", "3UTR"))
```

```
options(repr.plot.width=12, repr.plot.height=6)

# to plot them next to each other
all_tiles_df <- all_tiles_df %>%
  mutate(
    Adjusted_Bin = case_when(
      feature == "5UTR" ~ tile_index - 0.5,
      feature == "CDS" ~ tile_index + 19.5,
      feature == "3UTR" ~ tile_index + 39.5
    )
  )

# Calculate averages for the average plot
all_tiles_df_avg <- all_tiles_df %>%
  group_by(feature, Adjusted_Bin) %>%
  summarise(avg_percentage = mean(percentage, na.rm = TRUE), .groups = "drop")

# Averaged piRNA targeting across all genes with segmented x-axis
featureTargetingPlotTotal <- ggplot(all_tiles_df_avg, aes(x = Adjusted_Bin, y =
avg_percentage, color = feature)) +
  geom_line(linewidth = 1.5) +
  geom_vline(xintercept = c(20, 40), linetype = "dashed", color = "black") +
  labs(
    title = "top 250 mRNAs targeted by piRNAs derived from PG-as piCs",
    x = "Feature Segments (5' UTR, CDS, 3' UTR)",
    y = "Average Percentage of piRNAs Targeting the Gene"
  ) +
  theme_classic() +
  theme(
    legend.position = "bottom"
  )
featureTargetingPlotTotal

top250_wPGgenes <-
(targetingByPiRNAsSbO_sorted_total_tra[(targetingByPiRNAsSbO_sorted_total_tra$PGasToTopP
iC == "YES") & (targetingByPiRNAsSbO_sorted_total_tra$targetedRank <= 250),])$geneName
# mRNAs targeted by piRNAs derived from Pseudogenes
# Calculate averages for the average plot
all_tiles_df_avg_wPG <- all_tiles_df[all_tiles_df$gene_id %in% top250_wPGgenes,] %>%
  group_by(feature, Adjusted_Bin) %>%
  summarise(avg_percentage = mean(percentage, na.rm = TRUE), .groups = "drop")

# Averaged piRNA targeting across all genes with segmented x-axis
featureTargetingPlotTotal_wPG <- ggplot(all_tiles_df_avg_wPG, aes(x = Adjusted_Bin, y =
avg_percentage, color = feature)) +
  geom_line(linewidth = 1.5) +
  geom_vline(xintercept = c(20, 40), linetype = "dashed", color = "black") +
  labs(
    title = paste0(length(top250_wPGgenes), " mRNAs targeted by piRNAs derived from
Pseudogenes"),
    x = "Feature Segments (5' UTR, CDS, 3' UTR)",
    y = "Average Percentage of piRNAs Targeting the Gene"
  ) +
  theme_classic() +
  theme(
```

```
    legend.position = "bottom"
  )
featureTargetingPlotTotal_wPG
```



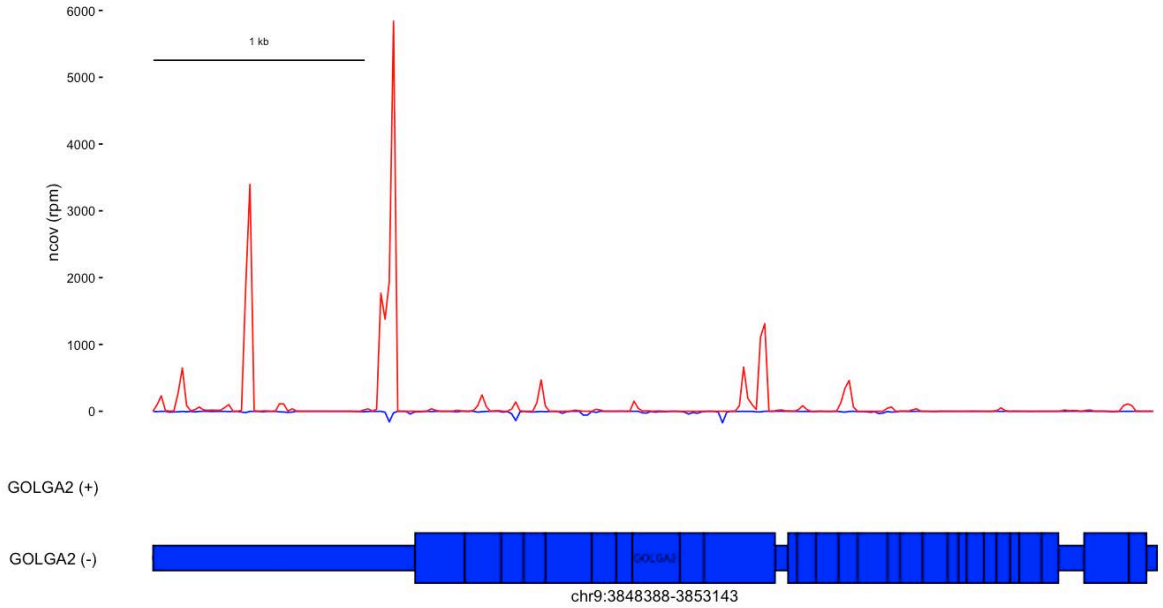
Coverage Plots (Extended Data Figure 4b,c)

Gene-targeting (shown in Custom Protein Coding Gene Transcriptome)

```
# GOLGA2 - Extended Data Figure 4b
options(repr.plot.width=12, repr.plot.height=6)
chr <- "chr9"
start <- 3848388
end <- 3853143
coord <- IRanges(start, end)
geneAnnotation_PCG$type <- geneAnnotation_PCG$feature

GOLGA2targeting<- allTracksPlotted(piRNAs_from_Bam = gr_alignments, chromosome = chr,
IRangesCoord=coord, gtfFiles=list(GOLGA2 = geneAnnotation_PCG), tilesWidth=20)
GOLGA2targetingFull <- GOLGA2targeting$plotCoverageTrack /
```

```
GOLGA2targeting$trackAll$gtfNum1$trackPlus / GOLGA2targeting$trackAll$gtfNum1$trackMinus
GOLGA2targetingFull
```

```
Normalizing to RPM
```

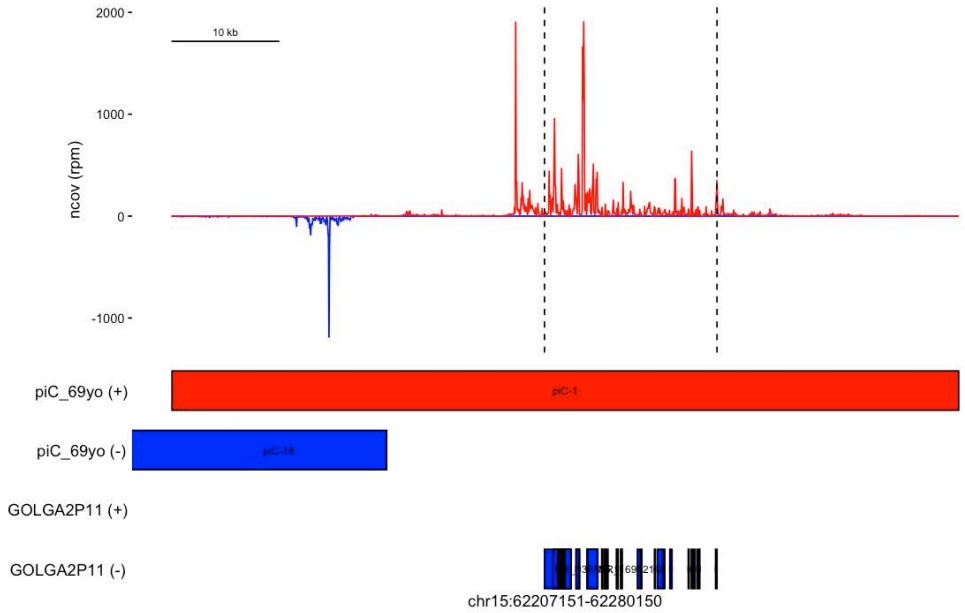


piRNA Cluster (piC) region as the origin of gene targeting

```
GOLGA2PG_dir <- "../data/annotations/RefSeqFor_chr15_62207201-62280100.gtf" #download
from UCSC Genome Browser
GOLGA2PG_grF <- rtracklayer::import(GOLGA2PG_dir)
GOLGA2PG_gr <- GOLGA2PG_grF[GOLGA2PG_grF$gene_id %in% c("NR_169521.2", "NR_136885.1")]
```

```
# GOLGA2 - Extended Data Figure 4c
chr <- as.character(seqnames(clusters_69yo[1]))
start <- start(clusters_69yo[1]) - 50
end <- end(clusters_69yo[1]) + 50
coord <- IRanges(start, end)

GOLGA2originPiC <- allTracksPlotted(piRNAs_from_Bam = gr_hg38, chromosome = chr,
IRangesCoord=coord, gtfFiles=list(piC_69yo = clusters_69yo, GOLGA2P11 = GOLGA2PG_gr),
tilesWidth=50, scaleWidthKB = 10)
t <- GOLGA2originPiC$plotCoverageTrack  +
    geom_vline(xintercept = min(start(GOLGA2PG_gr)), linetype = "dashed", color =
"black") +
    geom_vline(xintercept = max(end(GOLGA2PG_gr)), linetype = "dashed", color = "black")

GOLGA2originPiCFull <- t /
    GOLGA2originPiC$trackAll$gtfNum1$trackPlus /
GOLGA2originPiC$trackAll$gtfNum1$trackMinus /
    GOLGA2originPiC$trackAll$gtfNum2$trackPlus /
GOLGA2originPiC$trackAll$gtfNum2$trackMinus

GOLGA2originPiCFull
```

```
Normalizing to RPM
```



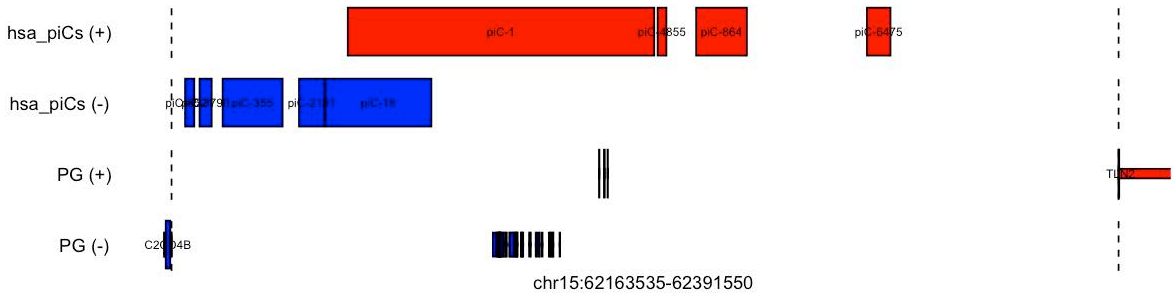
Evolutionary conserved pachytene piCs (Figure 4d)

Human

```
# GOLGA2
genes_names <- c("C2CD4B", "TLN2")
selRanges <- genes[genes$gene_name %in% genes_names]
chr <- as.character(unique(seqnames(selRanges)))
start <- min(start(selRanges))
end <- min(start(selRanges[selRanges$gene_name == "TLN2"]))+1000
coord <- IRanges(start, end)
GRanges(seqnames = chr, ranges = coord)

options(repr.plot.width=10, repr.plot.height=4)
GOLGA2_hsa <- allTracksPlotted(chromosome = chr, IRangesCoord=coord,
gtfFiles=list(hsa_piCs = clusters_69yo, PG = c(GOLGA2PG_grF, genes)), tilesWidth=100,
scaleWidthKB = 50)
piCsP <- GOLGA2_hsa$trackAll$gtfNum1$trackPlus +
    geom_vline(xintercept = max(end(selRanges[selRanges$gene_name == "C2CD4B"])),
linetype = "dashed", color = "black") +
    geom_vline(xintercept = min(start(selRanges[selRanges$gene_name == "TLN2"])),
linetype = "dashed", color = "black")
piCsM <- GOLGA2_hsa$trackAll$gtfNum1$trackMinus +
    geom_vline(xintercept = max(end(selRanges[selRanges$gene_name == "C2CD4B"])),
linetype = "dashed", color = "black") +
    geom_vline(xintercept = min(start(selRanges[selRanges$gene_name == "TLN2"])),
linetype = "dashed", color = "black")

genesP <- GOLGA2_hsa$trackAll$gtfNum2$trackPlus +
    geom_vline(xintercept = max(end(selRanges[selRanges$gene_name == "C2CD4B"])),
linetype = "dashed", color = "black") +
    geom_vline(xintercept = min(start(selRanges[selRanges$gene_name == "TLN2"])),
linetype = "dashed", color = "black")
genesM <- GOLGA2_hsa$trackAll$gtfNum2$trackMinus +
    geom_vline(xintercept = max(end(selRanges[selRanges$gene_name == "C2CD4B"])),
linetype = "dashed", color = "black") +
    geom_vline(xintercept = min(start(selRanges[selRanges$gene_name == "TLN2"])),
linetype = "dashed", color = "black")
```

```
GOLGA2hsaFull <- piCsP / piCsM / genesP / genesM
GOLGA2hsaFull
```

```
GRanges object with 1 range and 0 metadata columns:
      seqnames            ranges strand
         <Rle>         <IRanges>  <Rle>
  [1]    chr15 62163535-62391550      *
  -------
  seqinfo: 1 sequence from an unspecified genome; no seqlengths
```

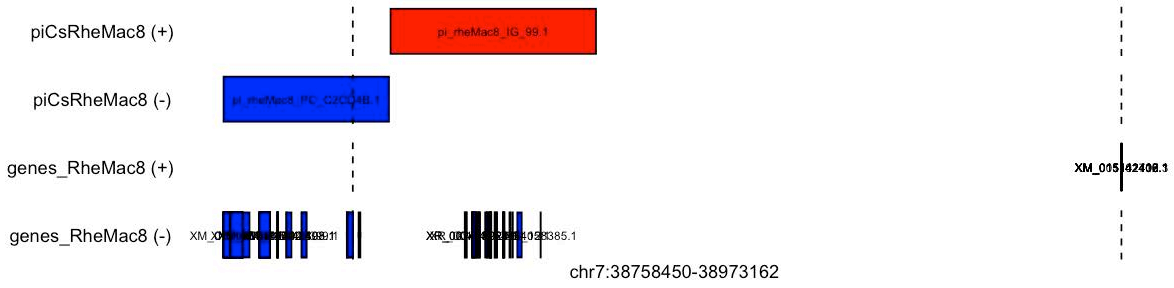


Rhesus macaque

```
#RHESUS, chr7:38,755,710-38,974,746

genes_rheMac8 <-
rtracklayer::import("../data/annotations/EvoComparison/rheMac8_GOLGA2PG.gtf") #download
from UCSC Genome Browser
genes_rheMac8$gene_name <- genes_rheMac8$gene_id
genes_rheMac8$type <- "exon"

# Published piRNA clusters from Yu et al. (Nat Commun, 2021):
https://doi.org/10.1038/s41467-020-20345-3
piCs_rheMac8 <- GRanges(
    seqnames = c("chr7", "chr7"),
    ranges = IRanges(c(38798200, 38758449), c(38847100, 38797823)),
    strand = c("+", "-"),
    type = c("CDS", "CDS"),
    gene_name = c("pi_rheMac8_IG_99.1", "pi_rheMac8_PC_C2CD4B.1"))


#XM_015142398.1 until XM_015142404.1 but XM_015142398.1 is the one matching ENSEMBL pred
= C2CD4B
#XM_001101705.3 = TLN2
genes_names <- c("XM_015142398.1", "XM_001101705.3")
selRanges <- genes_rheMac8[genes_rheMac8$gene_id %in% genes_names]
chr <- as.character(unique(seqnames(selRanges)))
start <- min(start(selRanges))
end <- min(start(selRanges[selRanges$gene_id == "XM_001101705.3"]))+1000
coord <- IRanges(start, end)
GRanges(seqnames = chr, ranges = coord)

GOLGA2_rheMac8 <- allTracksPlotted(chromosome = chr, IRangesCoord=coord,
gtfFiles=list(piCsRheMac8 = piCs_rheMac8, genes_RheMac8 = genes_rheMac8),
tilesWidth=100, scaleWidthKB = 50)
piCsP <- GOLGA2_rheMac8$trackAll$gtfNum1$trackPlus +
```

```
    geom_vline(xintercept = max(end(selRanges[selRanges$gene_name ==
"XM_015142398.1"])), linetype = "dashed", color = "black") +
    geom_vline(xintercept = min(start(selRanges[selRanges$gene_name ==
"XM_001101705.3"])), linetype = "dashed", color = "black")

piCsM <- GOLGA2_rheMac8$trackAll$gtfNum1$trackMinus +
    geom_vline(xintercept = max(end(selRanges[selRanges$gene_name ==
"XM_015142398.1"])), linetype = "dashed", color = "black") +
    geom_vline(xintercept = min(start(selRanges[selRanges$gene_name ==
"XM_001101705.3"])), linetype = "dashed", color = "black")

PGsP <- GOLGA2_rheMac8$trackAll$gtfNum2$trackPlus +
    geom_vline(xintercept = max(end(selRanges[selRanges$gene_name ==
"XM_015142398.1"])), linetype = "dashed", color = "black") +
    geom_vline(xintercept = min(start(selRanges[selRanges$gene_name ==
"XM_001101705.3"])), linetype = "dashed", color = "black")

PGsM <- GOLGA2_rheMac8$trackAll$gtfNum2$trackMinus +
    geom_vline(xintercept = max(end(selRanges[selRanges$gene_name ==
"XM_015142398.1"])), linetype = "dashed", color = "black") +
    geom_vline(xintercept = min(start(selRanges[selRanges$gene_name ==
"XM_001101705.3"])), linetype = "dashed", color = "black")

GOLGA2_rheMac8Full <- piCsP / piCsM / PGsP / PGsM
GOLGA2_rheMac8Full
```

```
GRanges object with 1 range and 0 metadata columns:
      seqnames            ranges strand
         <Rle>         <IRanges>  <Rle>
  [1]     chr7 38758450-38973162      *
  -------
  seqinfo: 1 sequence from an unspecified genome; no seqlengths
```

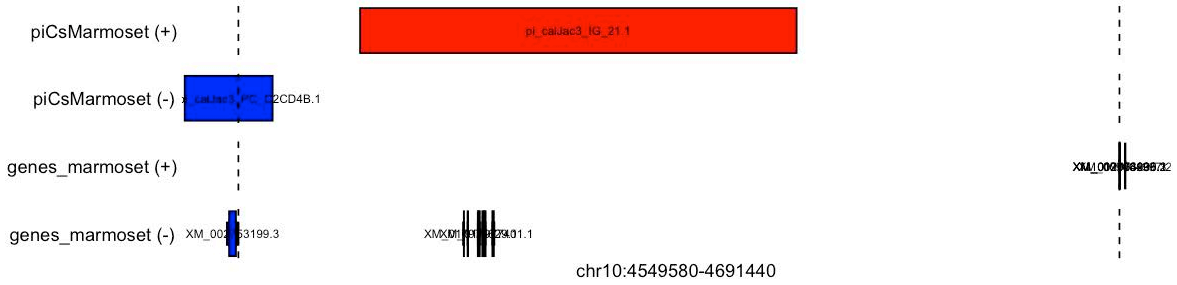


Marmoset

```
# MARMOSET, chr10:4,546,947-4,698,170
genes_calJac3 <-
rtracklayer::import("../data/annotations/EvoComparison/calJac3_GOLGA2PG.gtf")
genes_calJac3$gene_name <- genes_calJac3$gene_id

piCs_calJac3 <- GRanges(
    seqnames = c("chr10", "chr10"),
    ranges = IRanges(c(4542921, 4570600), c(4556809, 4639500)),
    strand = c("-", "+"),
    type = c("CDS", "CDS"),
```

```
    gene_name = c("pi_calJac3_PC_C2CD4B.1", "pi_calJac3_IG_21.1")
)


#XM_002753199.3 = C2CD4B
#XM_017976635.1 = TLN2
genes_names <- c("XM_002753199.3", "XM_017976635.1")
selRanges <- genes_calJac3[genes_calJac3$gene_id %in% genes_names]
chr <- as.character(unique(seqnames(selRanges)))
start <- min(start(selRanges))
end <- min(start(selRanges[selRanges$gene_id == "XM_017976635.1"]))+1000
coord <- IRanges(start, end)
GRanges(seqnames = chr, ranges = coord)

GOLGA2_calJac3 <- allTracksPlotted(chromosome = chr, IRangesCoord=coord,
gtfFiles=list(piCsMarmoset = piCs_calJac3, genes_marmoset = genes_calJac3),
tilesWidth=100, scaleWidthKB = 50)
piCsP <- GOLGA2_calJac3$trackAll$gtfNum1$trackPlus +
    geom_vline(xintercept = max(end(selRanges[selRanges$gene_name ==
"XM_002753199.3"])), linetype = "dashed", color = "black") +
    geom_vline(xintercept = min(start(selRanges[selRanges$gene_name ==
"XM_017976635.1"])), linetype = "dashed", color = "black")

piCsM <- GOLGA2_calJac3$trackAll$gtfNum1$trackMinus +
    geom_vline(xintercept = max(end(selRanges[selRanges$gene_name ==
"XM_002753199.3"])), linetype = "dashed", color = "black") +
    geom_vline(xintercept = min(start(selRanges[selRanges$gene_name ==
"XM_017976635.1"])), linetype = "dashed", color = "black")

genesP <- GOLGA2_calJac3$trackAll$gtfNum2$trackPlus +
    geom_vline(xintercept = max(end(selRanges[selRanges$gene_name ==
"XM_002753199.3"])), linetype = "dashed", color = "black") +
    geom_vline(xintercept = min(start(selRanges[selRanges$gene_name ==
"XM_017976635.1"])), linetype = "dashed", color = "black")

genesM <- GOLGA2_calJac3$trackAll$gtfNum2$trackMinus +
    geom_vline(xintercept = max(end(selRanges[selRanges$gene_name ==
"XM_002753199.3"])), linetype = "dashed", color = "black") +
    geom_vline(xintercept = min(start(selRanges[selRanges$gene_name ==
"XM_017976635.1"])), linetype = "dashed", color = "black")

GOLGA2_calJac3Full <- piCsP / piCsM / genesP / genesM
GOLGA2_calJac3Full
```

```
GRanges object with 1 range and 0 metadata columns:
      seqnames          ranges strand
         <Rle>       <IRanges>  <Rle>
  [1]    chr10 4549580-4691440      *
  -------
  seqinfo: 1 sequence from an unspecified genome; no seqlengths
```



Cow

```
#COW - bosTau8, chr10:47,794,442-47,959,240
genes_bosTau8_est <-
rtracklayer::import("../data/annotations/EvoComparison/bosTau8_GOLGA2PG_est.gtf") #
download from UCSC Genome Browser, cow expressed sequence tags (ESTs) in GenBank
genes_bosTau8_refSeq <-
rtracklayer::import("../data/annotations/EvoComparison/bosTau8_GOLGA2PG.gtf") # download
from UCSC Genome Browser, RefSeq
genes_bosTau8 <- c(genes_bosTau8_est, genes_bosTau8_refSeq)
genes_bosTau8$gene_name <- genes_bosTau8$gene_id

piCs_bosTau8 <- GRanges(
    seqnames = c("chr10", "chr10"),
    ranges = IRanges(c(47892700, 47931000), c(47930800, 47947700)),
    strand = c("-", "+"),
    type = c("CDS", "CDS"),
    gene_name = c("pi_bosTau8_IG_3.1", "pi_bosTau8_IG_4.1"))

# from genes_bosTau8: C2CD4B = NM_001046307
# from genes_bosTau8_est: TLN2 = e.g. EH375118

genes_names <- c("EH375118", "NM_001046307")
selRanges <- genes_bosTau8[genes_bosTau8$gene_id %in% genes_names]
chr <- as.character(unique(seqnames(selRanges)))
start <- max(end(selRanges[selRanges$gene_id == "EH375118"])) - 1000
end <- max(end(selRanges[selRanges$gene_id == "NM_001046307"]))
coord <- IRanges(start, end)

GOLGA2_bosTau8 <- allTracksPlotted(chromosome = chr, IRangesCoord=coord,
gtfFiles=list(piCsCow = piCs_bosTau8, genes_Cow = genes_bosTau8), tilesWidth=100,
scaleWidthKB = 50)
piCsP <- GOLGA2_bosTau8$trackAll$gtfNum1$trackPlus +
    geom_vline(xintercept = max(end(selRanges[selRanges$gene_name == "EH375118"])),
linetype = "dashed", color = "black") +
    geom_vline(xintercept = min(start(selRanges[selRanges$gene_name ==
"NM_001046307"])), linetype = "dashed", color = "black")

piCsM <- GOLGA2_bosTau8$trackAll$gtfNum1$trackMinus +
    geom_vline(xintercept = max(end(selRanges[selRanges$gene_name == "EH375118"])),
linetype = "dashed", color = "black") +
    geom_vline(xintercept = min(start(selRanges[selRanges$gene_name ==
"NM_001046307"])), linetype = "dashed", color = "black")
```

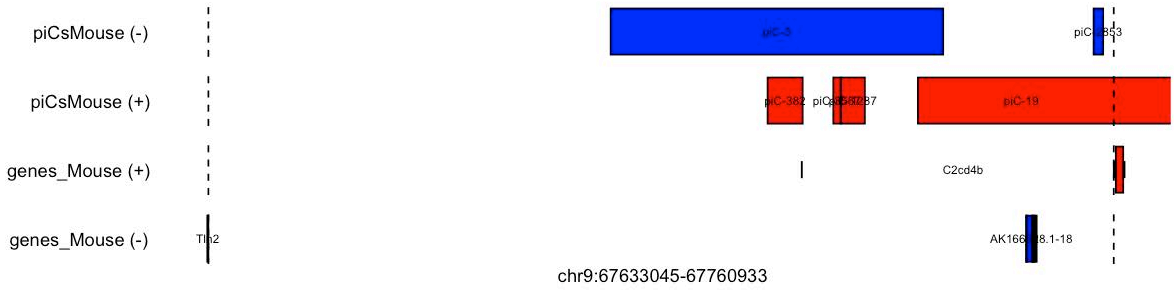
```
PGsP <- GOLGA2_bosTau8$trackAll$gtfNum2$trackPlus +
    geom_vline(xintercept = max(end(selRanges[selRanges$gene_name == "EH375118"])),
linetype = "dashed", color = "black") +
    geom_vline(xintercept = min(start(selRanges[selRanges$gene_name ==
"NM_001046307"])), linetype = "dashed", color = "black")

PGsM <- GOLGA2_bosTau8$trackAll$gtfNum2$trackMinus +
    geom_vline(xintercept = max(end(selRanges[selRanges$gene_name == "EH375118"])),
linetype = "dashed", color = "black") +
    geom_vline(xintercept = min(start(selRanges[selRanges$gene_name ==
"NM_001046307"])), linetype = "dashed", color = "black")

GOLGA2_bosTau8Full <- piCsP / piCsM / PGsP / PGsM
GOLGA2_bosTau8Full
```

Mouse

```
# MOUSE - chr9:67,555,573-67,763,784
genes_mm10 <-
rtracklayer::import("../data/annotations/mm10_collapsed_prioritizedCDS3UTR5UTR.gtf")
genes_mm10$gene_name <- genes_mm10$gene_id

#piRNA cluster coordinates from Konstantinidou et al. (2024) in Cell Reports and rank
load("../data/annotations/MILIclusters_pachytene.RData")
#MILI_prepach_regions_overl_genes
MILIclusters_pach <- MILIclusters$clusters
MILIclusters_pach <- MILIclusters_pach[order(-
MILIclusters_pach$all_reads_primary_alignments_FPM), ]
MILIclusters_pach$rankByAllReadsPrimaryAlignmentsFPM <- seq_along(MILIclusters_pach)
MILIclusters_pach$gene_name <- paste0("piC-",
MILIclusters_pach$rankByAllReadsPrimaryAlignmentsFPM)
MILIclusters_pach$type <- "CDS"

#load pseudogenes, download by UCSC
pseudogenes_mm10 <- rtracklayer::import("../data/annotations/mm10_retroGenesV6.gtf")
pseudogenes_mm10$gene_name <- pseudogenes_mm10$gene_id
```

```
# from genes_mm10: C2cd4b, Tln2
genes_names <- c("C2cd4b", "Tln2")
selRanges <- genes_mm10[genes_mm10$gene_id %in% genes_names]
chr <- as.character(unique(seqnames(selRanges)))
start <- max(end(selRanges[selRanges$gene_id == "Tln2"])) - 1000
```

```
end <- max(end(selRanges[selRanges$gene_id == "C2cd4b"]))
coord <- IRanges(start, end)

Spin1as_mm10 <- allTracksPlotted(chromosome = chr, IRangesCoord=coord,
gtfFiles=list(piCsMouse = MILIclusters_pach, genes_Mouse = c(pseudogenes_mm10,
genes_mm10)))

Spin1as_mm10M <- Spin1as_mm10$trackAll$gtfNum1$trackMinus +
    geom_vline(xintercept = max(end(selRanges[selRanges$gene_name == "Tln2"])), linetype
= "dashed", color = "black") +
    geom_vline(xintercept = 67759437, linetype = "dashed", color = "black") #start of
RefSeq NM_001081314.2

Spin1as_mm10P <- Spin1as_mm10$trackAll$gtfNum1$trackPlus +
    geom_vline(xintercept = max(end(selRanges[selRanges$gene_name == "Tln2"])), linetype
= "dashed", color = "black") +
    geom_vline(xintercept = 67759437, linetype = "dashed", color = "black") #start of
RefSeq NM_001081314.2

Spin1as_mm10PGsP <- Spin1as_mm10$trackAll$gtfNum2$trackPlus +
    geom_vline(xintercept = max(end(selRanges[selRanges$gene_name == "Tln2"])), linetype
= "dashed", color = "black") +
    geom_vline(xintercept = 67759437, linetype = "dashed", color = "black") #start of
RefSeq NM_001081314.2

Spin1as_mm10PGsM <- Spin1as_mm10$trackAll$gtfNum2$trackMinus +
    geom_vline(xintercept = max(end(selRanges[selRanges$gene_name == "Tln2"])), linetype
= "dashed", color = "black") +
    geom_vline(xintercept = 67759437, linetype = "dashed", color = "black") #start of
RefSeq NM_001081314.2

Spin1as_mm10Full <- Spin1as_mm10M / Spin1as_mm10P / Spin1as_mm10PGsP / Spin1as_mm10PGsM
Spin1as_mm10Full
```



Rat

```
#RAT - rn6, chr8:73,438,583-73,595,561
genes_rn6 <- rtracklayer::import("../data/annotations/EvoComparison/rn6_SPIN1PG.gtf")
genes_rn6$gene_name <- genes_rn6$gene_id

piCs_rn6 <- GRanges(
    seqnames = c("chr8", "chr8"),
    ranges = IRanges(c(73533100, 73572400), c(73572100, 73599100)),
    strand = c("-", "+"),
```

```
    type = c("CDS", "CDS"),
    gene_name = c("pi_rn6_IG_101.1", "pi_rn6_IG_102.1"))


# from genes_rn6: C2CD4B = XM_576426.6; TLN2 = e.g. XM_008766404.2
genes_names <- c("XM_576426.6", "XM_008766404.2")
selRanges <- genes_rn6[genes_rn6$gene_id %in% genes_names]
chr <- as.character(unique(seqnames(selRanges)))
start <- max(end(selRanges[selRanges$gene_id == "XM_008766404.2"])) - 1000
end <- min(start(selRanges[selRanges$gene_id == "XM_576426.6"])) + 1000
coord <- IRanges(start, end)

Spin1as_rn6 <- allTracksPlotted(chromosome = chr, IRangesCoord=coord,
gtfFiles=list(piCsRat = piCs_rn6, genes_Rat = genes_rn6), tilesWidth=100, scaleWidthKB =
50)
piCsP <- Spin1as_rn6$trackAll$gtfNum1$trackPlus +
    geom_vline(xintercept = min(start(selRanges[selRanges$gene_name == "XM_576426.6"])),
linetype = "dashed", color = "black") +
    geom_vline(xintercept = max(end(selRanges[selRanges$gene_name ==
"XM_008766404.2"])), linetype = "dashed", color = "black")

piCsM <- Spin1as_rn6$trackAll$gtfNum1$trackMinus +
    geom_vline(xintercept = min(start(selRanges[selRanges$gene_name == "XM_576426.6"])),
linetype = "dashed", color = "black") +
    geom_vline(xintercept = max(end(selRanges[selRanges$gene_name ==
"XM_008766404.2"])), linetype = "dashed", color = "black")

genesP <- Spin1as_rn6$trackAll$gtfNum2$trackPlus +
    geom_vline(xintercept = min(start(selRanges[selRanges$gene_name == "XM_576426.6"])),
linetype = "dashed", color = "black") +
    geom_vline(xintercept = max(end(selRanges[selRanges$gene_name ==
"XM_008766404.2"])), linetype = "dashed", color = "black")

genesM <- Spin1as_rn6$trackAll$gtfNum2$trackMinus +
    geom_vline(xintercept = min(start(selRanges[selRanges$gene_name == "XM_576426.6"])),
linetype = "dashed", color = "black") +
    geom_vline(xintercept = max(end(selRanges[selRanges$gene_name ==
"XM_008766404.2"])), linetype = "dashed", color = "black")

Spin1as_rn6Full <- piCsP / piCsM / genesP / genesM
Spin1as_rn6Full
```

For the schematic in Fig 4d, only the piRNA cluster, the neighboring genes (Tln2 and C2cd4b) as well as the Spin1/GOLGA2-Pseudogene are shown. Other gene annotations are removed to keep the schematic simple.

Sequence identity across the GOLGA2-associated piRNA locus in primates (Extended Data Figure 4e)

```
# Load genomes
library(BSgenome.Hsapiens.UCSC.hg38)
library(BSgenome.Mmulatta.UCSC.rheMac8)
library(BSgenome.Cjacchus.UCSC.calJac3)
```

```
# Human - GOLGA2 GENE, get sequence
hsa_GOLGA2gene <- genes[genes$gene_name %in% c("GOLGA2") & genes$type == "exon"]
hsa_GOLGA2gene_seq <- suppressWarnings(DNAStringSet(getSeq(BSgenome.Hsapiens.UCSC.hg38,
GRanges(seqnames = seqnames(hsa_GOLGA2gene[1]), ranges = IRanges(start =
min(start(hsa_GOLGA2gene)), end = max(end(hsa_GOLGA2gene)))))))
names(hsa_GOLGA2gene_seq) <- "hg38_GOLGA2"
```

```
# Human - full piRNA cluster with GOLGA2 PG, get sequence
hsa_piC_seq <- DNAStringSet(getSeq(BSgenome.Hsapiens.UCSC.hg38, clusters_69yo[1]))
names(hsa_piC_seq) <- "hsa_piC1"
```

```
# Macaque - full annotated piRNA cluster with GOLGA2 PG, get sequence
rheMac8_piC_seq <- DNAStringSet(getSeq(BSgenome.Mmulatta.UCSC.rheMac8, piCs_rheMac8[1]))
names(rheMac8_piC_seq) <- "rheMac8_piC_IG99"
```

```
# Marmoset - full annotated piRNA cluster with GOLGA2 PG, get sequence
calJac3_piC_seq <- DNAStringSet(getSeq(BSgenome.Cjacchus.UCSC.calJac3, piCs_calJac3[2]))
names(calJac3_piC_seq) <- "calJac3_piC_IG21"
```

```
# Marmoset - GOLGA2 GENE, get sequence
calJac3_genes <-
rtracklayer::import("../data/annotations/EvoComparison/calJac3_NCBI_RefSeq.gtf")
calJac3_GOLGA2gene <- calJac3_genes[calJac3_genes$gene_id == "XM_009003501.2" &
calJac3_genes$type == "exon"]
calJac3_GOLGA2gene_seq <- DNAStringSet(getSeq(BSgenome.Cjacchus.UCSC.calJac3,
GRanges(seqnames = seqnames(calJac3_GOLGA2gene[1]), ranges = IRanges(start =
min(start(calJac3_GOLGA2gene)), end = max(end(calJac3_GOLGA2gene))))))
names(calJac3_GOLGA2gene_seq) <- "calJac3_GOLGA2"
```

```
# Combine seqs and save as fasta
all_seqs <- c(hsa_GOLGA2gene_seq, hsa_piC_seq, rheMac8_piC_seq, calJac3_piC_seq,
calJac3_GOLGA2gene_seq)
Biostrings::writeXStringSet(all_seqs,
"../data/annotations/EvoComparison/GOLGA2_HSA_RHEMAC8_CALJAC3.fasta", format = "fasta")
```

BASH

```
# minimap2, all-versus-all alignment
minimap2 -x asm20 -c --eqx -D -P --dual=no \
    ../data/annotations/EvoComparison/GOLGA2_HSA_RHEMAC8_CALJAC3.fasta \
    ../data/annotations/EvoComparison/GOLGA2_HSA_RHEMAC8_CALJAC3.fasta \
```

```
    >
../data/annotations/EvoComparison/GOLGA2_HSA_RHEMAC8_CALJAC3_x_asm20_c_eqx_D_P_dualNo.pa
f
```

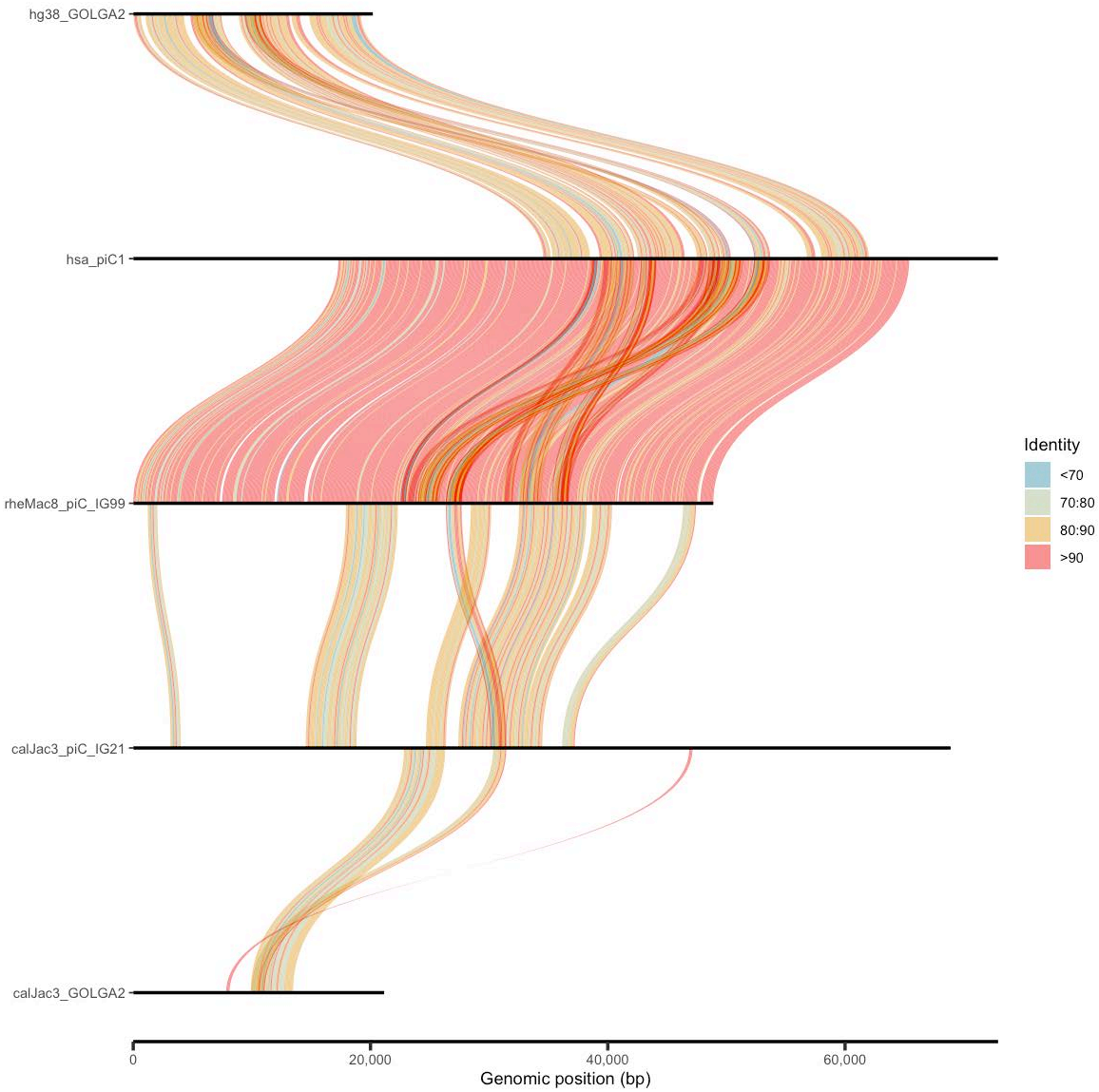
```
[M::mm_idx_gen::0.006*1.88] collected minimizers
[M::mm_idx_gen::0.013*2.42] sorted minimizers
[M::main::0.013*2.41] loaded/built the index for 5 target sequence(s)
[M::mm_mapopt_update::0.014*2.35] mid_occ = 50
[M::mm_idx_stat] kmer size: 19; skip: 10; is_hpc: 0; #seq: 5
[M::mm_idx_stat::0.014*2.30] distinct minimizers: 35632 (88.19% are singletons);
average occurrences: 1.172; average spacing: 5.556; total length: 232053
[M::worker_pipeline::0.454*2.02] mapped 5 sequences
[M::main] Version: 2.30-r1287
[M::main] CMD: ../../../../../../../../../Documents/minimap2/minimap2 -x asm20 -
c --eqx -D -P --dual=no ../data/annotations/EvoComparison/GOLGA2_HSA_RHEMAC8_CAL
JAC3.fasta ../data/annotations/EvoComparison/GOLGA2_HSA_RHEMAC8_CALJAC3.fasta
[M::main] Real time: 0.456 sec; CPU: 0.919 sec; Peak RSS: 0.763 GB
```

R

In case you switched your kernel, rerun the *Prep*-section of this notebook first

```
options(repr.plot.width=10, repr.plot.height=10)
# read paf file (output of minimap2), using SVbyEye package
paf.table <- readPaf(
    paf.file =
"../data/annotations/EvoComparison/GOLGA2_HSA_RHEMAC8_CALJAC3_x_asm20_c_eqx_D_P_dualNo.p
af",
    include.paf.tags = TRUE
)

# Create plot with percent identity as color
plt_fullPiC <- plotAVA(paf.table = paf.table, color.by = "direction", binsize = 100,
        perc.identity.breaks = c(70, 80, 90), seqnames.order= c("hg38_GOLGA2",
"hsa_piC1", "rheMac8_piC_IG99", "calJac3_piC_IG21", "calJac3_GOLGA2"))

plt_fullPiC
```

```
[readPaf] Loading PAF file: ../data/annotations/EvoComparison/GOLGA2_HSA_RHEMAC8
_CALJAC3_x_asm20_c_eqx_D_P_dualNo.paf
 ... 0.01s

[pafToBins] Binning PAF alignments, binsize=100bp
 ... 8.32s
```


hg38_GOLGA2
hsa_piC1
rheMac8_piC_IG99
calJac3_piC_IG21
calJac3_GOLGA2
0
20,000
40,000
60,000
Genomic position (bp)
Identity
<70
70:80
80:90
>90