wind: wORKFLOW FOR PiRNAs AnD BEYONd

Computational workflow for the creation of Gene transfer format file with small-RNA sequences, GRCm38

Constantinos Yeles (Konstantinos Geles)

Thu Oct 29 2020

Contents

Introduction

With the intent to annotate and quantify small-RNA sequence data (and in particular piRNA) derived from Next-Generation Sequencing, a workflow has been implemented. For the generation of annotation files and results widely used tools of alignment, annotation, quantification and differential expression algorithms have been utilized. Although the workflow is focused particularly on piRNAs (as is our main subject of research) with slight modifications can be applied to all small-RNA categories of interest.

To make it more versatile and reproducible, we adopted the *containerization approach* as the software deployment is fast, efficient, and potentially bug-free. It can be used in various operating systems with only requirements the installation of the docker engine and have some minimum requirements of processing power and RAM to run the most memory demanding tools.

Materials and Methods

The workflow has been primarily carried out on a Linux server, but it can be used easily on a Windows or Mac OS machine as long as changes have been done to appropriate functions/operations.

The workflow utilizes *Bash* and *R* scripting for various operations. For the application of the workflow, the following tools have been used:

- *Rstudio* for R scripting,
- *STAR* for alignment,
- *Samtools* for various modifications and extraction of reads from resulted aligned files,
- *FastQC* for quality control,
- *Cutadapt* for adapter trimming,
- *bedtools* for bam to bed manipulation,
- *Salmon* for transcript-level quantification,
- *featureCounts* for transcript-level quantification

Databases that have been used:

- *piRNABank* for piRNA sequences,
- *RNAcentral* for smallRNA sequences

Workflow

1. Acquisition and Preprocessing of the small ncRNA sequences

i. Downloading the files for the generation of a Gene transfer format (gtf)

piRNA sequences for mouse were downloaded from piRNABank to enrich in piRNA sequences the gtf file, and small-RNA genome coordinates (bed files) from RNACentral have been acquired

```
# all the files and folders for the workflow are created in the working directory
# plus the results of the analysis
docker run --rm -ti -v $(pwd):/home/my_data  congelos/sncrna_workflow

mkdir -p my_data/mouse_data/GRCm38

# piRNAbank sequences
wget http://pirnabank.ibab.ac.in/downloads/all/mouse_all.zip -O my_data/mouse_data/mouse_all.zip

unzip -d my_data/mouse_data/  my_data/mouse_data/mouse_all.zip && rm my_data/mouse_data/mouse_all.zip

# RNAcentral genome coordinates
wget http://ftp.ebi.ac.uk/pub/databases/RNAcentral/current_release//genome_coordinates/bed/mus_musculus

# GRCm38 fasta for STAR index
wget ftp://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_mouse/release_M25/GRCm38.primary_assembly.genome

pigz -d my_data/mouse_data/GRCm38/GRCm38.primary_assembly.genome.fa.gz
```

ii. Preprocessing of the piRNAbank file

The fasta file from piRNAbank has U character instead of T in the sequences, so changes have been made using sed

```
sed 's/U/T/g' my_data/mouse_data/mouse_pir.txt > my_data/mouse_data/pirnaBank_mouse.fasta
# exit docker container
exit
```

iii. Run docker and Load libraries

```
docker run --rm -v $(pwd):/home/0 -p 8787:8787 -e PASSWORD=12345 \
-e USER=$UID congelos/rocker_tidyverse_plus_de_pckages
# we prefer to work on Rstudio to perform everything on R otherwise R on
# bash can be used directly
```

```
suppressPackageStartupMessages({
  library('tidyverse')
  library('data.table')
  library('plyranges')
  library("BSgenome.Mmusculus.UCSC.mm10")
})
```

iv. Remove duplicated sequences

```
pirnaB_mm8 <- Biostrings::readDNAStringSet("mouse_data/pirnaBank_mouse.fasta")
pirnaB_mm8 %>% length() ## >[1] 1399813

# remove duplicate sequences-----
pirnaB_mm8 <- pirnaB_mm8[!duplicated(pirnaB_mm8)]
pirnaB_mm8 %>% length() ## >[1] 39986

# clean the names----
names(pirnaB_mm8) <- names(pirnaB_mm8) %>%
  str_remove("\\|M.+") %>%
  str_replace("\\|gb\\|","_")

# write the fasta ----
pirnaB_mm8 %>%
  Biostrings::writeXStringSet("mouse_data/pirnaB_mm8_removed_duplicates.fa" )
```

```
# exit docker container
exit
```

v. Align piRNA sequences to mouse genome

Afterwards, alignment of piRNA sequences to the mouse genome utilizing STAR aligner has been performed. Then, samtools has been used to export the sequences in fasta format

```
docker run --rm -ti -v $(pwd):/home/my_data  congelos/sncrna_workflow

# create index
STAR --runMode genomeGenerate --genomeDir my_data/mouse_data/GRCm38 --genomeFastaFiles my_data/mouse_da

mkdir my_data/mouse_data/piRNABank_mouse_mm10

# align the piRNABank sequences
STAR --genomeDir my_data/mouse_data/GRCm38 --genomeLoad LoadAndKeep \
--readFilesIn "my_data/mouse_data/pirnaB_mm8_removed_duplicates.fa"  \
--runThreadN 6 --alignIntronMax 0 --outSAMattributes NH HI NM MD \
--outFilterMultimapNmax 100 --outReadsUnmapped Fastx --outFilterMismatchNmax 0 \
--outFilterMatchNmin 16 --outFileNamePrefix "my_data/mouse_data/piRNABank_mouse_mm10/piBnk_mm10_"

# sort the sam file
samtools sort -O bam -o my_data/mouse_data/piRNABank_mouse_mm10/piBnk_mm10_sorted.bam -@ 6 \
my_data/mouse_data/piRNABank_mouse_mm10/piBnk_mm10_Aligned.out.sam

# BAM to fasta format
samtools fasta -F 4 -@ 8 \
my_data/mouse_data/piRNABank_mouse_mm10/piBnk_mm10_sorted.bam > my_data/mouse_data/piRNABank_mouse_mm10/

# BAM to bed format
bedtools bamtobed < my_data/mouse_data/piRNABank_mouse_mm10/piBnk_mm10_sorted.bam > my_data/mouse_data/p

exit
```

2. Unification of pirnaBANK sequences and RNAcentral ncRNA sequences

Run docker, load libraries

```
docker run --rm -v $(pwd):/home/0 -p 8787:8787 -e PASSWORD=12345 \
-e USER=$UID congelos/rocker_tidyverse_plus_de_pckages
```

```
suppressPackageStartupMessages({
  library('tidyverse')
  library('data.table')
  library('plyranges')
  library("BSgenome.Mmusculus.UCSC.mm10")
})
```

i. RNAcentral. import RNAcentral file

```
sRNA <- read_bed("mouse_data/mus_musculus.GRCm38.bed.gz") %>%
  select("sRNA_id" = name, "gene_type" = NA.1, "source" = NA.2) %>%
  mutate(type = "exon")

sInfo <- Seqinfo(genome="mm10")
seqlevels(sInfo) <- seqlevels(sRNA)
seqinfo(sRNA)<- sInfo
```

ii. RNAcentral. filtering for sequences smaller than 100 bps

```
tr_sRNA <- sRNA %>%
  as_tibble() %>% # [1,065,469] genomic ranges / [180,780] sRNA_ids
  filter(width < 100) %>% # [962,711] GR / [94,653] sRNA_ids
  mutate(sRNA_id = str_remove(sRNA_id,"_10090")) %>%
  as_granges() %>%
  # keep info about the standard chromosomes
  keepStandardChromosomes(pruning.mode = "coarse") %>% # [962,711] -> [947,408] GRs / [94,653] -> [94,6
  # remove the duplicated entries from RNAcentral
  as_tibble() %>%
  unite(col = "seq_s",seqnames:strand, sep = "_") %>%
  distinct(seq_s, .keep_all = TRUE) %>% # remove dupl GRanges [947,408] -> [929,168] GRs / [94,644] ->
  separate(col = seq_s,into = c("seqnames","start","end","width","strand"),
    sep = "_") %>%
  mutate(start = as.numeric(start),
    end = as.numeric(end),
    width = as.numeric(width)) %>%
  as_granges()

biotypes <- tr_sRNA %>%
  as_tibble() %>%
  select(sRNA_id,gene_type) %>%
  distinct(sRNA_id, .keep_all = T)
```

iii. RNAcentral. keep sequence information

```
transcripts_mouse <- Views(BSgenome.Mmusculus.UCSC.mm10, tr_sRNA)

# search for duplicated sequences ----
fasta_tr_ms <- DNAStringSet(transcripts_mouse)
names(fasta_tr_ms) <- mcols(transcripts_mouse)$sRNA_id
fasta_tr_ms <- fasta_tr_ms[sort(fasta_tr_ms@ranges@NAMES)]

fasta_tr_ms_tbl <- fasta_tr_ms  %>%
  as.character() %>%
  enframe(name = "tr_mm10" ,value = "mm10") %>%
  left_join(biotypes, by = c("tr_mm10" = "sRNA_id"))

fasta_tr_ms_tbl %>%
  distinct(tr_mm10, mm10, gene_type, .keep_all = TRUE) %>%
  filter(duplicated(mm10)) %>%
  count(mm10,sort = TRUE)

#  duplicates between sequences and genomic locations(GRanges)-----
## make a tibble with all GR, seq and ids
transcripts_GR <- transcripts_mouse %>%
    as_granges() %>%
    as_tibble() %>%
    unite(col = "seq_RCent", seqnames:strand, sep = "_")

## find unique pairs of seq and GR

uniq_seq <- transcripts_GR  %>%
  distinct(dna, .keep_all = TRUE) %>%
  arrange(dna) %>%
  mutate(sRNA_id2 = str_c(sRNA_id,"_GR_",seq_RCent)) %>%
  select(dna,sRNA_id2)

transcripts_GR <- transcripts_GR %>%
  left_join(uniq_seq)
```

iv. piRNABank. import the aligned to genome fasta

```
piRNAbank_mm10 <- Biostrings::readDNAStringSet("mouse_data/piRNABank_mouse_mm10/piBnk_mm10_sorted.fasta
piRNAbank_mm10_tib <- piRNAbank_mm10 %>% # 39,380 sequences
  as.character() %>%
  enframe(value = "seq_piBn") %>%
  mutate(sRNA_type ="piRNA",
    bpairs_piR = str_length(seq_piBn)) %>%
  arrange(desc(bpairs_piR))
```

v. piRNAbank. make Genomic Ranges and remove duplicates from GRCm38

```r
# piRNABank. import the granges and change it -----
  piRNAbank_mm10_ranges <- read_bed("mouse_data/piRNABank_mouse_mm10/piBnk_mm10_sorted.bed") %>%
  as_tibble() %>%
  arrange(desc(width)) %>% # [55,905] GRs / [39,380] sequences
  filter(width < 100) %>% # [55,905] -> [55,903] GRs / [39,380] sequences
  as_granges() %>%
  keepStandardChromosomes(pruning.mode = "coarse")# [55,903] -> [54,677] GRs / [39,380]seq

transcripts_pi_mm10 <- Views(BSgenome.Mmusculus.UCSC.mm10, piRNAbank_mm10_ranges) %>%
  as_granges() %>%
  keepStandardChromosomes(pruning.mode = "coarse") %>%
  as_tibble()

# we need to apply a second width filter at 68
# as we know that piRNAs are ~32 base pairs
transcripts_pi_mm10 <- transcripts_pi_mm10 %>% filter(width < 68) # [54,677] -> [54,676] GRs / [39,380]
transcripts_pi_mm10 %>% count(name) %>% nrow #> 39380 piRNAs from piRNABANK

# checking sequences of alignments with lower length
sequen_pi_false <- transcripts_pi_mm10 %>%
  as_tibble() %>%
  left_join(piRNAbank_mm10_tib) %>%
  arrange(desc(width)) %>%
  mutate(sequences_true = (dna == seq_piBn)) %>%
  filter(sequences_true == FALSE) %>%
  unite(col = "seq_s",seqnames:strand, sep = "_")

# piRNABank. removing duplicated GR ----
piRNAbank_mm10_ranges %>%
  as_tibble() %>%
  unite(col = "seq_s",seqnames:strand, sep = "_") %>%
  count(seq_s) %>%
  filter(n > 1) %>%
  .$seq_s %>%
  map(~sequen_pi_false %>%
  filter(seq_s == .x)) %>%
  bind_rows()

transcripts_pi_mm10_clean <- transcripts_pi_mm10 %>%
  as_tibble() %>%
  left_join(piRNAbank_mm10_tib) %>%
  arrange(desc(width)) %>%
  mutate(sequences_true = (dna == seq_piBn)) %>%
  filter(sequences_true == TRUE) %>% # [54,676] -> [54,014] GRs / [39,380] -> [39,380] sequences
  select(-score, -seq_piBn, -bpairs_piR, -sequences_true) %>%
  unite(col = "seq_piBNK",seqnames:strand, sep = "_")

transcripts_pi_mm10_clean %>% count(name) %>% nrow #> 39380 piRNAs final piRNABANK
#> 54,014 Genomic ranges

# sanity checks ----
```

```r
concated_mm10_piBAnk %>%
  filter(!duplicated(dna)) %>%
  select(sRNA_id, seq_id) %>%
  arrange(sRNA_id)

tr_test <- transcripts_mouse %>% as_granges()

concated_mm10_piBAnk %>%
  filter(!duplicated(dna), !is.na(name))  %>%
  select(seq_RCent,seq_id,sRNA_id)

transcripts_pi_mm10_clean %>%  filter(name == "mmu_piR_013613_DQ691404")

tr_test %>% filter(sRNA_id == "URS00004A4FA0") %>%
  select(gene_type,dna) %>% arrange(seqnames)

## checking for the duplicated sequences
concated_mm10_piBAnk %>%
  filter(duplicated(dna)) %>%
  distinct(dna, .keep_all = T) %>%
  select(seq_RCent,seq_id)

concated_mm10_piBAnk %>% filter(sRNA_id == "URS00003AC5A0")
transcripts_pi_mm10_clean %>%  filter(name == "mmu_piR_014163_DQ692225")
tr_test %>% filter(sRNA_id == "URS00003AC5A0")
```

vi. RNAcentral. + piRNABank. make annotation tibble

```r
mm10_piBAnk_RCent <- transcripts_GR %>%
  left_join(piRNAbank_mm10_tib, by = c("dna" = "seq_piBn"))
# check gene_types
mm10_piBAnk_RCent %>%
  filter(is.na(gene_type))
mm10_piBAnk_RCent %>%
  filter(is.na(name)) %>%
  count(gene_type)
mm10_piBAnk_RCent %>%
  filter(!is.na(name)) %>%
  count(gene_type)
mm10_piBAnk_RCent %>%
  filter(!is.na(name)) %>%
  filter(!gene_type == "piRNA") %>%
  count(gene_type)

mm10_piBAnk_RCent%>%
  filter(gene_type != "piRNA",sRNA_type == "piRNA") %>%
  count(gene_type)
# in case that in piRNAbank a pirna is the same but with different type in RNAcentral
# we will keep the gene type of RNAcentral
concated_mm10_piBAnk <- mm10_piBAnk_RCent  %>%
  mutate(
    seq_id = case_when(
      gene_type == "piRNA" & sRNA_type == "piRNA" ~ name,
      gene_type != "piRNA" & sRNA_type == "piRNA" ~ str_c(sRNA_id2, "_", name),
      is.na(sRNA_type) ~ sRNA_id2
    ),
    source = case_when(
      sRNA_type == "piRNA" ~ str_c(source, ",piRNABank"),
      is.na(sRNA_type) ~ source,

    )
  )

concated_mm10_piBAnk %>%
  filter(!is.na(name)) %>%
  filter(!gene_type == "piRNA") %>%
  count(gene_type)
# sanity checks ----
## checking for the NA values, should be only true
(concated_mm10_piBAnk %>%
    filter(is.na(name)) %>% .$sRNA_id2 ==
    concated_mm10_piBAnk %>%
    filter(is.na(name)) %>% .$seq_id
  ) %>% table

## checking for the miRNA values, should be 16 miRNA false
 (concated_mm10_piBAnk %>% filter(gene_type == "miRNA") %>% .$sRNA_id2 ==
     concated_mm10_piBAnk %>% filter(gene_type == "miRNA") %>% .$seq_id
   ) %>% table

## function for all gene_types
fun_unm <- function(x){
  (concated_mm10_piBAnk %>%
      filter(gene_type == x) %>%
      .$sRNA_id2 ==
      concated_mm10_piBAnk %>%
      filter(gene_type == x) %>%
      .$seq_id
    ) %>% table
}
## checking for all gene_types, should be only true except piRNAs, miRNA and misc_RNA
concated_mm10_piBAnk %>%
  count(gene_type) %>%
  .$gene_type %>% set_names(.) %>%
  map(~fun_unm(.x))

## checking for the piRNA values
is.na(concated_mm10_piBAnk$seq_id) %>% table

## checking for duplicates
concated_mm10_piBAnk %>%
  filter(duplicated(seq_id)) %>%
  arrange(name)

concated_mm10_piBAnk %>%
  filter(duplicated(name),!is.na(name)) %>%
  arrange(name)

concated_mm10_piBAnk %>%
  filter(duplicated(sRNA_id2),!is.na(sRNA_id2)) %>%
  arrange(name)

dupl_seqs <- concated_mm10_piBAnk %>%
  filter(duplicated(dna)) %>%
  arrange(name)

fasta_tr_ms_tbl %>%
  filter(mm10 %in% dupl_seqs$dna)
```

vii. RNAcentral + piRNABank. generation of GRanges

```r
concated_mm10_piBAnk  # df with combined sequences piRNAbank+RNAcentral
transcripts_pi_mm10_clean # has all alignments from piRNAbank
tr_sRNA # has the ranges with less than 100 bp from RNAcentral

c_piBNK_RCent <- concated_mm10_piBAnk %>%
  full_join(transcripts_pi_mm10_clean, by = c("dna",
    "name", "sRNA_type", "seq_RCent" =  "seq_piBNK")) %>%
  select(seq_RCent,
    sRNA_id,
    name,
    seq_id,
    gene_type,
    sRNA_type,
    everything()) %>%
  mutate(source =
      case_when(
        is.na(source) ~ "piRNA_BANK",
        TRUE ~ source),
    gene_type =
      case_when(
        is.na(gene_type) ~ sRNA_type,
        TRUE ~ gene_type),
    seq_id =
      case_when(
        is.na(seq_id) ~ name,
        TRUE ~ seq_id),
    type =
      case_when(
        is.na(type) ~ "exon",
        TRUE ~ type)
  )


c_piBNK_RCent %>% count(sRNA_id, sort = T)

c_piBNK_RCent %>% count(name, sort = T)
c_piBNK_RCent %>% count(seq_id, sort = T)

c_piBNK_RCent %>% count(gene_type, sort = T)

c_piBNK_RCent %>% count(source, sort = T)

c_piBNK_RCent %>% count(type, sort = T)

c_piBNK_RCent %>% count(dna, sort = T)

c_piBNK_RCent %>% count(sRNA_id2, sort = T)

c_piBNK_RCent %>% count(bpairs_piR, sort = T)

# final Genomic ranges -----
c_piBNK_RCent_GR <- c_piBNK_RCent %>%
  select(-name, -sRNA_type, -bpairs_piR) %>%
  rename( dna = "seq_RNA") %>%
  separate(col = seq_RCent,into = c("seqnames",
    "start","end","width","strand"), sep = "_") %>%
  mutate(start = as.numeric(start),
    end = as.numeric(end),
    width = as.numeric(width)) %>%
  as_granges

c_piBNK_RCent_GR %>%
  as_tibble() %>%
  arrange(desc(width)) %>%
  count(width, gene_type, sort =T) %>%
  write_tsv("mouse_data/piRNAbank_mouse_RNACentral_bp_info.txt")

# possible "clusters" of piRNA -----
c_piBNK_RCent_GR %>%
  filter(gene_type == "piRNA") %>%
  plyranges::reduce_ranges_directed() %>%
  as_tibble() %>%
  filter(width > 40) %>%
  arrange(desc(width)) %>%
  count(width) %>% view

# testing of sequences-----
piRNAbank_rCentral_seqs <- Views(BSgenome.Mmusculus.UCSC.mm10, c_piBNK_RCent_GR)

piRNAbank_rCentral_seqs %>%
  as_granges() %>%
  as_tibble() %>% mutate( is_it_TR = (seq_RNA == dna)) %>%
  filter(is_it_TR == FALSE)# should be 0

# final objects to export-----
piRNAbank_rCentral_fasta <- DNAStringSet(piRNAbank_rCentral_seqs)
names(piRNAbank_rCentral_fasta) <- mcols(piRNAbank_rCentral_seqs)$seq_id

piRNAbank_rCentral_fasta <- piRNAbank_rCentral_fasta[!duplicated(piRNAbank_rCentral_fasta)]
```

3. Save the results to fasta and gtf format

```
piRNAbank_rCentral_fasta %>%
  Biostrings::writeXStringSet("mouse_data/ncRNA_transcripts_100bp_RNA_Central_piRNAbank_mm10.fa")

gtf_piB_RCentr <- piRNAbank_rCentral_seqs %>%
  as_granges() %>%
  as_tibble() %>%
  dplyr::rename("gene_id" = seq_id) %>%
  select(-dna) %>%
  as_granges()

sInfo <- Seqinfo(genome="mm10")
seqlevels(sInfo) <- seqlevels(gtf_piB_RCentr)
seqinfo(gtf_piB_RCentr)<- sInfo

gtf_piB_RCentr %>%
  as_granges() %>%
  write_gff2("mouse_data/ncRNA_transcripts_100bp_RNA_Central_piRNAbank_mm10.gtf")

# to import use:
# fasta_tr_hg38 <- Biostrings::readDNAStringSet("mouse_data/ncRNA_transcripts_100bp_RNA_Central_piRNAba
# gtf_piB_RCentr <- read_gff2("mouse_data/ncRNA_transcripts_100bp_RNA_Central_piRNAbank_mm10.gtf")
```

Following this workflow the files for annotation and quantification of small-RNA samples have been prepared. Afterwards, the steps on the *GSM3535476_workflow.pdf* could be followed to perform pre-processing of the samples, alignment, quantification and calculation of transcript abundances.

4. Provide extra information regarding genomic locations, genes, transcripts, for the gtf

Load libraries

```
suppressPackageStartupMessages({
library('TxDb.Mmusculus.UCSC.mm10.knownGene')
library('org.Mm.eg.db')
library('bumphunter')
library('BiocParallel')
library('stats')
})
```

import regions of transcripts

```
genes <- annotateTranscripts(TxDb.Mmusculus.UCSC.mm10.knownGene, annotation="org.Mm.eg.db") %>%
  keepStandardChromosomes(pruning.mode="coarse") %>% arrange(seqnames)

piRNAbank_rCentral_gtf <- read_gff2("mouse_data/ncRNA_transcripts_100bp_RNA_Central_piRNAbank_mm10.gtf"
identical(genes %>% seqlevels(), piRNAbank_rCentral_gtf %>% seqlevels())

piRNAbank_rCentral_gtf %>% length()

map(piRNAbank_rCentral_gtf %>% seqlevels() %>% purrr::set_names(),
    ~piRNAbank_rCentral_gtf %>%
        filter(seqnames == .x) %>%
        length()) %>% bind_rows() %>%
  pivot_longer(cols = chr1:chrM) %>%
  arrange(desc(value))

# we will parallelize per chr.

 if(.Platform$OS.type == "windows") {
mt_param <- SnowParam()
 } else{
mt_param <- MulticoreParam()
 }
# we will work with 10 workers
mt_param <- MulticoreParam(workers = 8)

# simple function which takes lists of Granges and the chromosome
# name to select from each list
matchGenes_fun <- function(our_Grs, genes_GRs){
  suppressPackageStartupMessages({
    library('dplyr')
    library('bumphunter')
  })
  message("working on matchGenes")
  matchGenes(our_Grs, genes_GRs,
    type = "any", promoterDist = 2500,
    skipExons = FALSE, verbose = TRUE) %>% as_tibble()
}

genes_chr <- map(genes %>% seqlevels() %>% purrr::set_names(),
  ~genes %>% filter(seqnames == .x))

gen_test <- genes_chr[c("chrM","chrY")]

piR_chr <- map(piRNAbank_rCentral_gtf %>%
    seqlevels() %>%
    purrr::set_names(), ~piRNAbank_rCentral_gtf %>%
        filter(seqnames == .x))

piR_test <- piR_chr[c("chrM","chrY")]

res_chr <- bpmapply(matchGenes_fun,
  piR_chr, genes_chr, USE.NAMES=TRUE, SIMPLIFY = FALSE,
    BPREDO=list(), BPPARAM = mt_param)

res_chr <- bind_rows(res_chr) %>%
  bind_cols(as_tibble(piRNAbank_rCentral_gtf)) %>%
  dplyr::select(name:subjectHits, gene_id,
    gene_type, sRNA_id, source, seq_RNA) %>%
  write_tsv("mouse_data/gene_regions_piRNAbank_rCentral.txt")
```

5. Find multimapping piRNAs

```
multi_test <- piRNAbank_rCentral_gtf %>%
    plyranges::select(gene_id, seq_RNA, gene_type) %>% join_overlap_inner_directed(plyranges::select(pi
  arrange(seqnames)

multi_test %>%
  filter(gene_type.x == "piRNA",
    !gene_id.x == gene_id.y ) %>%
  as_tibble() %>%
  count(gene_type.x, gene_type.y, sort = T) %>%
  write_tsv("genomic_locations_stats_multi.txt")

piRNAbank_rCentral_gtf %>%
  filter(gene_type == "piRNA") %>%
  plyranges::select(-c(score, phase, source, type)) %>%
  as_tibble() %>%
  unite(col = "seq_s",seqnames:strand, sep = "_") %>%
  count(gene_id, sort = T) %>% write_tsv("genomic_locations_stats_multi_piRNA.txt")
```

6. Find how many piRNAs are in common and uncommon in piRNABank and RNAcentral in the new gtf

```
c_piBNK_RCent %>% distinct(seq_id, .keep_all = T)

c_piBNK_RCent %>% distinct(seq_id, .keep_all = T) %>% filter(is.na(sRNA_type),gene_type == "piRNA")

c_piBNK_RCent %>% distinct(seq_id, .keep_all = T) %>% filter(!is.na(sRNA_type),is.na(sRNA_id),gene_type

c_piBNK_RCent %>% distinct(seq_id, .keep_all = T) %>% filter(!is.na(name),!is.na(sRNA_id), gene_type==
```

7. Find which smallRNAs are inside Trasposable Elements

We have downloaded a gtf file with the information about genomic regions of Transposable Elements for human genome: http://labshare.cshl.edu/shares/mhammelllab/www-data/TEtranscripts/TE_GTF/ more precisely: GRCm38_Ensembl_rmsk_TE.gtf.gz

```
TEs <- read_gff2("GRCm38_Ensembl_rmsk_TE.gtf.gz") %>%
  plyranges::select("TE_gene_id" = gene_id, "TE_transcript_id" = transcript_id,
   "TE_family_id" = family_id, "TE_class_id" = class_id) %>%
  keepStandardChromosomes(pruning.mode = "coarse") %>%
  arrange(seqnames)

piRNAbank_rCentral_gtf %>%
  plyranges::select(gene_id, sRNA_id,gene_type, seq_RNA) %>%
  find_overlaps_directed(TEs) %>%
  write_gff2("TEs_piRNAbank_rCentral.gtf")

piRNAbank_rCentral_gtf %>%
  join_overlap_left_directed( piRNAbank_rCentral_gtf %>%
  find_overlaps_directed(TEs)) %>% length()

piRNAbank_rCentral_gtf %>%
  find_overlaps_directed(TEs) %>%
  plyranges::reduce_ranges_directed() %>% length()
```

8. indexes for STAR and Salmon

```
docker run --rm -ti -v $(pwd):/home/my_data  congelos/sncrna_workflow

STAR --runMode genomeGenerate --genomeDir my_data/mouse_data/GRCm38 --genomeFastaFiles my_data/mouse_da

# following the instructions for salmon decoy aware indexing
# https://combine-lab.github.io/alevin-tutorial/2019/selective-alignment/
grep "^>" <my_data/mouse_data/GRCm38/GRCm38.primary_assembly.genome.fa | cut -d " " -f 1 > my_data/mous

sed -i.bak -e 's/>//g' my_data/decoys.txt

cat my_data/ncRNA_transcripts_100bp_RNA_Central_piRNAbank_mm10.fa my_data/ncRNA_genome_100bp_RNA_Centra

exit

# run the docker
docker run --rm -it -v $(pwd):/home/my_data combinelab/salmon

# create the index
## *remove manually the spike-ins chrs from decoys
salmon index -t my_data/mouse_data/gentrome_GRCm38_piRNAbank_Central.fa -d my_data/mouse_data/decoys.tx
exit
```