wind: wORKFLOW FOR PiRNAs AnD BEYONd

Computational workflow for the creation of Gene transfer format file with smallRNA sequences

Constantinos Yeles(Konstantinos Geles)

Thu Oct 15 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 with the intention to generate files utilizing widely used tools of alignment, annotation, quantification and differential expression algorithms. Although it 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 modification 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

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

piRNA sequences for human were downloaded from piRNABank to enrich in piRNA sequences the gtf file, and smallRNA 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/human_data/hg38

# downloading the piRNAbank sequences
wget http://pirnabank.ibab.ac.in/downloads/all/human_all.zip -O my_data/human_data/human_all.zip

unzip -d my_data/human_data/  my_data/human_data/human_all.zip

# downloading the RNAcentral genomic coordinates
wget ftp://ftp.ebi.ac.uk/pub/databases/RNAcentral/current_release/\
genome_coordinates/bed/homo_sapiens.GRCh38.bed.gz -O my_data/human_data/homo_sapiens.GRCh38.bed.gz

# GRCh38 fasta for STAR index
wget ftp://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_human/release_34/GRCh38.primary_assembly.genome.

pigz -d my_data/human_data/hg38/GRCh38.primary_assembly.genome.fa.gz
```

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/human_data/human_pir.txt > my_data/human_data/pirnaBank_human.fasta

# exit docker container
exit
```

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 could be used directly
```

```
suppressPackageStartupMessages({
  library('tidyverse')
  library('data.table')
```

```
  library('plyranges')
  library("BSgenome.Hsapiens.UCSC.hg38")
})
```

Remove duplicated sequences

In the piRNABank fasta exist duplicated sequences so we remove them

```
# import the fasta  ----
pirnaB_hg19 <- Biostrings::readDNAStringSet("human_data/pirnaBank_human.fasta")
pirnaB_hg19 %>% length() ## >[1] 667944

# remove duplicated sequences----
pirnaB_hg19 <- pirnaB_hg19[!duplicated(pirnaB_hg19)]
pirnaB_hg19 %>% length() ## >[1] 23439

# clean the names----
names(pirnaB_hg19) <- names(pirnaB_hg19) %>%
  str_remove("\\|H.+") %>%
  str_replace("\\|gb\\|","_")

# write the fasta ----
pirnaB_hg19 %>%
  Biostrings::writeXStringSet("human_data/piRNAbank_hg19_removed_duplicates.fa" )
```

```
# exit docker container
exit
```

Align piRNA sequences to human genome

Afterwards, alignment of piRNA sequences to the human genome utilizing STAR aligner has been performed and then samtools used for the extraction of sequences in fasta format

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

STAR --runMode genomeGenerate --genomeDir my_data/human_data/hg38 --genomeFastaFiles my_data/human_data/

mkdir my_data/piRNABank_human_hg38

STAR --genomeDir hg38/ --genomeLoad LoadAndKeep \
--readFilesIn "my_data/human_data/piRNAbank_hg19_removed_duplicates.fa"  \
--runThreadN 10 --alignIntronMax 0 --outSAMattributes NH HI NM MD \
--outFilterMultimapNmax 100 --outReadsUnmapped Fastx --outFilterMismatchNmax 0 \
--outFilterMatchNmin 16 --outFileNamePrefix "my_data/piRNABank_human_hg38/piBnk38_"

samtools sort -O bam -o my_data/piRNABank_human_hg38/piBnk38_sorted.bam -@ 10 \
my_data/piRNABank_human_hg38/piBnk38_Aligned.out.sam

samtools fasta -F 4 -@ 8 \
my_data/piRNABank_human_hg38/piBnk38_sorted.bam > my_data/piRNABank_human_hg38/piBnk38_sorted.fasta
```

```
bedtools bamtobed < my_data/piRNABank_human_hg38/piBnk38_sorted.bam > my_data/piRNABank_human_hg38/piBnk

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.Hsapiens.UCSC.hg38")
})
```

1. piRNABank. import the aligned to genome fasta

```
piRNAbank_hg38 <- Biostrings::readDNAStringSet("piRNABank_human_hg38/piBnk38_sorted.fasta")
piRNAbank_hg38_tib <- piRNAbank_hg38 %>%
  as.character() %>%
  enframe(value = "seq_piBn") %>%
  mutate(sRNA_type ="piRNA",
    bpairs_piR = str_length(seq_piBn)) %>%
  arrange(desc(bpairs_piR))
```

2. RNAcentral. import RNAcentral file

```
sRNA <- read_bed("human_data/homo_sapiens.GRCh38.bed.gz") %>%
  select("sRNA_id" = name, "gene_type" = NA.1, "source" = NA.2) %>%
  mutate(type = "exon")
sInfo <- Seqinfo(genome="hg38")
seqlevels(sInfo) <- seqlevels(sRNA)
seqinfo(sRNA)<- sInfo
```

3. RNAcentral. filtering for sequences smaller than 100 bps

```
tr_sRNA <- sRNA %>%
  as_tibble() %>%
  filter(width < 100) %>%
  mutate(sRNA_id = str_remove(sRNA_id,"_9606")) %>%
  as_granges() %>%
  # keep info about the standard chromosomes
```

```
  keepStandardChromosomes(pruning.mode = "coarse") %>% # [162.958] -> [160.980]
  # remove the duplicated entries from RNAcentral
  as_tibble() %>%
  unite(col = "seq_s",seqnames:strand, sep = "_") %>%
  distinct(seq_s, .keep_all = TRUE) %>% # remove dupl GRanges [160.980] -> [153.043]
  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)
```

4. RNAcentral. keep sequence information

```
transcripts_human <- Views(BSgenome.Hsapiens.UCSC.hg38, tr_sRNA)

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

fasta_tr_hs_tbl <- fasta_tr_hs %>%
  as.character() %>%
  enframe(name = "tr_hg38" ,value = "hg38") %>%
  left_join(biotypes, by = c("tr_hg38" = "sRNA_id"))

fasta_tr_hs_tbl %>%
  distinct(tr_hg38, hg38, gene_type, .keep_all = TRUE) %>%
  filter(duplicated(hg38)) %>%
  count(hg38,sort = TRUE)

###  duplicates between sequences and genomic locations(GRanges)-----
# make a tibble with all GR, seq and ids
transcripts_GR <- transcripts_human %>%
    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)
```

5. RNAcentral. + piRNABank. tibble

```
hg38_piBAnk_RCent <- transcripts_GR %>%
  left_join(piRNAbank_hg38_tib, by = c("dna" = "seq_piBn"))

# check gene_types
hg38_piBAnk_RCent %>%
  filter(is.na(name)) %>%
  count(gene_type)
hg38_piBAnk_RCent %>%
  filter(!is.na(name)) %>%
  count(gene_type,sRNA_type)

concated_hg38_piBAnk <- hg38_piBAnk_RCent  %>%
  mutate(
    seq_id = case_when(
      is.na(gene_type) ~ name,
      gene_type != "piRNA" ~ sRNA_id2,
      is.na(sRNA_type) ~ sRNA_id2,
      TRUE ~ name
    )
  )
# sanity checks ----
## checking for the NA values, should be only true
(concated_hg38_piBAnk %>%
    filter(is.na(name)) %>% .$sRNA_id2 ==
    concated_hg38_piBAnk %>%
    filter(is.na(name)) %>% .$seq_id
  ) %>% table

## checking for the miRNA values, should be only true
 (concated_hg38_piBAnk %>% filter(gene_type == "miRNA") %>% .$sRNA_id2 ==
     concated_hg38_piBAnk %>% filter(gene_type == "miRNA") %>% .$seq_id
   ) %>% table

## function for all gene_types
fun_unm <- function(x){
  (concated_hg38_piBAnk %>%
      filter(gene_type == x) %>%
      .$sRNA_id2 ==
      concated_hg38_piBAnk %>%
      filter(gene_type == x) %>%
      .$seq_id
    ) %>% table
}
## checking for all gene_types, should be only true except piRNAs
concated_hg38_piBAnk %>%
  count(gene_type) %>%
  .$gene_type %>% set_names(.) %>%
```

```
  map(~fun_unm(.x))

## checking for the piRNA values
is.na(concated_hg38_piBAnk$seq_id) %>% table

## checking for duplicates
concated_hg38_piBAnk %>%
  filter(duplicated(seq_id)) %>%
  arrange(name)

concated_hg38_piBAnk %>%
  filter(duplicated(name),!is.na(name)) %>%
  arrange(name)

concated_hg38_piBAnk %>%
  filter(duplicated(sRNA_id2),!is.na(sRNA_id2)) %>%
  arrange(name)

dupl_seqs <- concated_hg38_piBAnk %>%
  filter(duplicated(dna)) %>%
  arrange(name)

fasta_tr_hs_tbl %>%
  filter(hg38 %in% dupl_seqs$dna)
```

6. piRNAbank. make Genomic Ranges and remove duplicates from hg38

```
# piRNABank. import the granges and change it -----
  piRNAbank_hg38_ranges <- read_bed("piRNABank_human_hg38/piBnk38_sorted.bed") %>%
  as_tibble() %>%
  arrange(desc(width)) %>%
  filter(width < 100) %>%
  as_granges() %>%
  keepStandardChromosomes(pruning.mode = "coarse")

transcripts_pi_hg38 <- Views(BSgenome.Hsapiens.UCSC.hg38, piRNAbank_hg38_ranges) %>%
  as_granges() %>%
  keepStandardChromosomes(pruning.mode = "coarse") %>%
  as_tibble()

# we need to apply a second width filter at 69
# as we know that piRNAs are ~32 base pairs
transcripts_pi_hg38 <- transcripts_pi_hg38 %>% filter(width < 69)
transcripts_pi_hg38 %>% count(name) %>% nrow #> 23120 piRNAs from piRNABANK

# checking sequences of alignments with lower length
sequen_pi_false <- transcripts_pi_hg38 %>%
  as_tibble() %>%
  left_join(piRNAbank_hg38_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_hg38_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_hg38_clean <- transcripts_pi_hg38 %>%
  as_tibble() %>%
  left_join(piRNAbank_hg38_tib) %>%
  arrange(desc(width)) %>%
  mutate(sequences_true = (dna == seq_piBn)) %>%
  filter(sequences_true == TRUE) %>%
  select(-score, -seq_piBn, -bpairs_piR, -sequences_true) %>%
  unite(col = "seq_piBNK",seqnames:strand, sep = "_")

transcripts_pi_hg38_clean %>% count(name) %>% nrow #> 23116 piRNAs final piRNABANK
#> 44,557 Genomic ranges

# sanity checks ----
concated_hg38_piBAnk %>%
  filter(!duplicated(dna)) %>%
  select(sRNA_id,seq_id) %>%
  arrange(sRNA_id)

tr_test <- transcripts_human %>% as_granges()

concated_hg38_piBAnk %>% filter(sRNA_id == "URS0000000096")
transcripts_pi_hg38_clean %>%  filter(name == "hsa_piR_009796_DQ583192")

tr_test %>% filter(sRNA_id == "URS0000000096")

## checking for the duplicated sequences
concated_hg38_piBAnk %>%
  filter(duplicated(dna)) %>%
  select(seq_RCent,seq_id)

concated_hg38_piBAnk %>% filter(sRNA_id == "URS00001B5714")
transcripts_pi_hg38_clean %>%  filter(name == "hsa_piR_011289_DQ585240")
tr_test %>% filter(sRNA_id == "URS00001B5714")
```

7. RNAcentral + piRNABank. generation of Granges

```
concated_hg38_piBAnk  # df with combined sequences piRNAbank+RNAcentral
transcripts_pi_hg38_clean # has all alignments from piRNAbank
tr_sRNA # has the ranges with less than 100 bp from RNAcentral
```

```
c_piBNK_RCent <- concated_hg38_piBAnk %>%
  full_join(transcripts_pi_hg38_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",
        !is.na(sRNA_type) ~ str_c("piRNA_BANK,",source),
        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)
  )


names(c_piBNK_RCent)

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("piRNAbank_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.Hsapiens.UCSC.hg38, 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("ncRNA_transcripts_100bp_RNA_Central_piRNAbank_hg38.fa")

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

sInfo <- Seqinfo(genome="hg38")
```

```
seqlevels(sInfo) <- seqlevels(gtf_piB_RCentr)
seqinfo(gtf_piB_RCentr)<- sInfo

gtf_piB_RCentr %>%
  as_granges() %>%
  write_gff2("ncRNA_transcripts_100bp_RNA_Central_piRNAbank_hg38.gtf")

# to import use:
# fasta_tr_hg38 <- Biostrings::readDNAStringSet("ncRNA_transcripts_100bp_RNA_Central_piRNAbank_hg38.fa"
# gtf_piB_RCentr <- read_gff2("ncRNA_transcripts_100bp_RNA_Central_piRNAbank_hg38.gtf")
```

Following this workflow the files for annotation and quantification of smallRNA samples have been prepared. Afterwards, the steps on the *GSM3535476_workflow.pdf* could be followed to perform preprocessing 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.Hsapiens.UCSC.hg38.knownGene')
library('org.Hs.eg.db')
library('bumphunter')
library('BiocParallel')
library('stats')
})
```

import regions of transcripts

```
genes <- annotateTranscripts(TxDb.Hsapiens.UCSC.hg38.knownGene, annotation="org.Hs.eg.db") %>%
  keepStandardChromosomes(pruning.mode="coarse") %>% arrange(seqnames)

piRNAbank_rCentral_gtf <- read_gff2("ncRNA_transcripts_100bp_RNA_Central_piRNAbank_hg38.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))
# chr15 has the most values of GRs with 15522
# 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 = 10)

# 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("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 which smalRNAs 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: GRCh38_GENCODE_rmsk_TE.gtf.gz

http://labshare.cshl.edu/shares/mhammelllab/www-data/TEtranscripts/TE_GTF/GRCh38_GENCODE_rmsk_TE.gtf.gz

```
TEs <- read_gff2("GRCh38_GENCODE_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()
```

7. Optional if there are spike-ins sequences

```
spike <- read_tsv("spike-ins.txt", col_names = c("names", "seq_RNA"))
fasta_gen_hg38 <- Biostrings::readDNAStringSet("human_data/hg38/GRCh38.primary_assembly.genome.fa")

spikes_Fasta <- spike$seq_RNA %>%
  DNAStringSet(start = rep(1,nrow(spike)), end = str_length(spike$seq_RNA)) %>%
  setNames(spike$names)

DNAStringSetList(spikes_Fasta,fasta_gen_hg38) %>%
  unlist() %>%
  Biostrings::writeXStringSet("ncRNA_genome_Spike_ins_100bp_RNA_Central_piRNAbank_hg38.fa")
```

```
fasta_tr_hg38 <- Biostrings::readDNAStringSet("ncRNA_transcripts_100bp_RNA_Central_piRNAbank_hg38.fa")

DNAStringSetList(spikes_Fasta,fasta_tr_hg38) %>%
  unlist() %>%
  Biostrings::writeXStringSet("ncRNA_transcripts_Spike_ins_100bp_RNA_Central_piRNAbank_hg38.fa")

gtf_piB_RCentr_spikes <- spike %>%
  dplyr::rename("seqnames" = names) %>%
  mutate(
  start = 1,
  end = str_length(seq_RNA),
  width = str_length(seq_RNA),
  strand = "+",
  gene_id = seqnames,
  type = "exon",
  source = "spike_in",
  gene_type = "spike_in"
) %>%
  as_granges() %>%
  bind_ranges(piRNAbank_rCentral_gtf) %>%
  write_gff2("ncRNA_transcripts_Spike_ins_100bp_RNA_Central_piRNAbank_hg38.gtf")

# file for histograms of piRNAs
gtf_piB_RCentr_spikes %>%
  as_tibble() %>%
  filter(gene_type %in% c("piRNA","spike_in")) %>%
  plyranges::select(gene_id, seq_RNA, gene_type) %>%
  distinct(seq_RNA , .keep_all = TRUE) %>%
  write_tsv("piRNAs_hist.txt")
```

indexes for STAR and Salmon with Spike-ins

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

STAR --runMode genomeGenerate --genomeDir my_data/human_data/hg38_spike_ins --genomeFastaFiles my_data/r

# following the instructions for salmon decoy aware indexing
# https://combine-lab.github.io/alevin-tutorial/2019/selective-alignment/
grep "^>" <my_data/ncRNA_genome_Spike_ins_100bp_RNA_Central_piRNAbank_hg38.fa | cut -d " " -f 1 > my_dat

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

cat my_data/ncRNA_transcripts_Spike_ins_100bp_RNA_Central_piRNAbank_hg38.fa my_data/ncRNA_genome_Spike_i

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/gentrome_piRNAbank_Central.fa -d my_data/decoys.txt -i my_data/ncRNA_Central_piB
exit
```

piRNA histograms

```
## filter for reads of 15-49 bases
for file in my_data/spike_ins/star_results/*/*_sorted.bam;
do
where_to_save=‘dirname ${file}‘;
regex=‘basename ${file}‘;
samp="${regex%%.trimmed_sorted.bam}";
echo "Processing sample ${samp} start: $(date)";
samtools view -h -@ 6 ${file} | awk ’length($10) > 14 && length($10) < 50 || $1 ~ /^@/’ | samtools view
echo "end:$(date)";
done
## find length of reads from STAR - FeatureCounts
cut -f1 my_data/piRNAs_hist.txt > my_data/piRNA_ids.txt

for file in my_data/spike_ins/star_results/*.featureCounts.bam;
do
where_to_save=‘dirname ${file}‘;
regex=‘basename ${file}‘;
samp="${regex%%.trimmed_sorted.bam.featureCounts.bam}";
echo "Processing sample ${samp} start: $(date)";
samtools view -@ 6 ${file} | awk ’BEGIN{FS=OFS="\t"}{print length($10),$18}’| sed ’s/XT:Z://g’ | grep -
echo "end:$(date)";
done

## find length of reads from salmon
for file in my_data/spike_ins/quants/*_sorted.bam;
do
where_to_save=‘dirname ${file}‘;
regex=‘basename ${file}‘;
samp="${regex%%.trimmed*}";
echo "Processing sample ${samp} start: $(date)";
samtools view  -@ 6 ${file} | awk ’BEGIN{FS=OFS="\t"}{print $1,length($10),$3}’| sort -k1,1 | bedtools
echo "end:$(date)";
done


samtools view -@ 6 ${file} | awk ’BEGIN{FS=OFS="\t"}{print length($10),$3}’| grep -F -w -f my_data/piRN
```

ggplot for histograms

```
library(tidyverse)
piRNAs_hist <- read_tsv("piRNAs_hist.txt")
# gtf_piB_RCentr <- read_gff2("ncRNA_transcripts_100bp_RNA_Central_piRNAbank_hg38.gtf") %>%
  as_tibble() %>%
  select(gene_id,gene_type) %>%
  distinct(gene_id, .keep_all = T)
# FC and salmon files -------
hist_files_fc <- list.files(path ="spike_ins/star_results",
                            pattern = "49_hist.txt",
```

```
                          recursive = TRUE, full.names = TRUE)

hist_files_salmon <- list.files(path ="spike_ins/quants",
                          pattern = "_hist_allRNA.txt",
                          recursive = TRUE, full.names = TRUE)

test1 <- read_tsv(hist_files_fc[1], col_names = c("Reads", "Length", "sncRNA"))

gtf_piB_RCentr <- gtf_piB_RCentr %>%
  add_case(gene_id= "SS_22", gene_type = "piRNA")

no_piRNA_reg_ex <- gtf_piB_RCentr %>%
  as_tibble() %>%
  filter(!gene_type == "piRNA") %>%
  distinct(gene_id) %>%
  .$gene_id %>%
  str_c(collapse = "|")

itest1 <- test1 %>%
  mutate(Length = as_factor(Length))

# featurecounts facet hist-----
pdf(str_glue("histograms_piRNA_reads_facets_fc.pdf"))
map(hist_files_fc, ~read_tsv(.x, col_names = c("Reads", "Length", "sncRNA")) %>%
  mutate(Length = as_factor(Length))  %>%
  mutate(Alignment = if_else(str_detect(sncRNA, no_piRNA_reg_ex),
                             true = "Multimapped",
                             false = "Unique" )) %>%

  group_by(Length,Alignment) %>%
  summarise( Reads = sum(Reads)) %>%
  ggplot() +
  geom_bar(mapping = aes(x = Length, y = Reads), stat = "identity")+
  scale_y_continuous(labels = scales::comma)+
  theme_minimal()+
  facet_grid(Alignment ~ .)+
  ggtitle(hist_files_fc[1] %>% basename %>% str_remove("_hist_pirna.txt"))
)
dev.off()

## pick only spike_ins----
spike_reg_ex <- piRNAs_hist %>%
  filter(gene_type == "spike_in") %>%
  .$gene_id %>%
  str_c(collapse = "|") %>%
  set_names("spike_ins")

piRNA_reg_ex <- piRNAs_hist %>%
  filter(gene_type == "piRNA") %>%
  .$gene_id %>%
  str_c(collapse = "|") %>%
  set_names("piRNA")
```

```
miRNA_reg_ex <- gtf_piB_RCentr %>%
  as_tibble() %>%
  filter(gene_type == "miRNA") %>%
  distinct(gene_id) %>%
  .$gene_id %>%
  str_c(collapse = "|") %>%
  set_names("miRNA")

pdf(str_glue("histograms_spike_ins_reads_Salmon.pdf"))

map(hist_files_salmon,~read_tsv(.x, col_names = c("Reads", "Length", "sncRNA")) %>%
  mutate(Length = as_factor(Length)) %>%
  filter(str_detect(sncRNA,spike_reg_ex)) %>%
  group_by(Length) %>%
  summarise(Reads = sum(Reads)) %>%
  ggplot() +
  geom_bar(mapping = aes(x = Length, y = Reads), stat = "identity")+
  scale_y_continuous(labels = scales::comma)+
  theme_minimal()+
  ggtitle(.x %>% basename %>% str_replace("allRNA.txt","spike_ins"))
)
  dev.off()

pdf(str_glue("histograms_piRNA_reads_FC_filtered.pdf"))

map(hist_files_fc, ~read_tsv(.x, col_names = c("Reads", "Length", "sncRNA")) %>%
  mutate(Length = as_factor(Length)) %>%
  filter(str_detect(sncRNA,piRNA_reg_ex)) %>%
  group_by(Length) %>%
  summarise(Reads = sum(Reads)) %>%
  ggplot() +
  geom_bar(mapping = aes(x = Length, y = Reads), stat = "identity")+
  scale_y_continuous(labels = scales::comma)+
  theme_minimal()+
  ggtitle(.x %>% basename %>% str_replace("allRNA.txt","piRNA"))
)
  dev.off()

pdf(str_glue("histograms_miRNA_reads_Salmon.pdf"))

map(hist_files_salmon,~read_tsv(.x, col_names = c("Reads", "Length", "sncRNA")) %>%
  mutate(Length = as_factor(Length)) %>%
  filter(str_detect(sncRNA,miRNA_reg_ex)) %>%
  group_by(Length) %>%
  summarise(Reads = sum(Reads)) %>%
  ggplot() +
  geom_bar(mapping = aes(x = Length, y = Reads), stat = "identity")+
  scale_y_continuous(labels = scales::comma)+
  theme_minimal()+
  ggtitle(.x %>% basename %>% str_replace("allRNA.txt","miRNA"))
)
  dev.off()
```

```
pdf(str_glue("histograms_all_RNA_reads_Salmon.pdf"))

map(hist_files_salmon,~read_tsv(.x, col_names = c("Reads", "Length", "sncRNA")) %>%
  mutate(Length = as_factor(Length)) %>%
  #filter(str_detect(sncRNA,miRNA_reg_ex)) %>%
  group_by(Length) %>%
  summarise(Reads = sum(Reads)) %>%
  ggplot() +
  geom_bar(mapping = aes(x = Length, y = Reads), stat = "identity")+
  scale_y_continuous(labels = scales::comma)+
  theme_minimal()+
  ggtitle(.x %>% basename %>% str_replace("allRNA.txt","all_RNA"))+
  coord_flip()
)
  dev.off()


## piRNA reads ------
reads_piRNA <- read_tsv("reads_piRNA.txt",
                        col_names = c("Read","Length","sRNAs","read_sequence", "Sigar"))

reads_piRNA %>% count(Sigar)
reads_piRNA %>% count(Length)

piRNA_reads <- reads_piRNA %>%
  select(Read, sRNAs, read_sequence, Length) %>%
  separate(sRNAs, str_c("V",1:2),
           extra = "merge", fill = "right", sep = ",") %>%
  filter(is.na(V2)) %>%
  select(-V2) %>%
  filter(str_detect(V1, piRNA_reg_ex))

piRNA_reads %>%
  mutate(Length = as_factor(Length)) %>%
  group_by(Length) %>%
  #summarise(Read) %>%
  ggplot() +
  geom_bar(mapping = aes(x = Length))+
  scale_y_continuous(labels = scales::comma)+
  theme_minimal()+
  ggtitle("COLO205_dil_A_NT_1_piRNA_reads" %>% basename %>% str_replace("allRNA.txt","all_RNA"))+
  coord_flip()


key_mIrna_pIrna <- gtf_piB_RCentr %>%
  as_tibble() %>%
  distinct(gene_id, .keep_all = T) %>%
  select(gene_id,seq_RNA,gene_type)

test_mut_reads <- reads_piRNA %>%
  #head(1000) %>%
  filter(str_detect(sRNAs, miRNA_reg_ex)) %>%
  select(Read, sRNAs, read_sequence) %>%
```

```
  separate(sRNAs, str_c("V",1:12),
           extra = "merge", fill = "right", sep = ",") %>%
  pivot_longer(cols = starts_with("V"),
               names_to = "alignment", values_to = "sRNAs", values_drop_na = T) %>%
  left_join(key_mIrna_pIrna, by = c("sRNAs" = "gene_id"))

test_mut_reads %>% filter(gene_type %in% c("miRNA", "piRNA"))
test_mut_reads %>%
  group_by(Read) %>%

# different way -----
mutate_all(~replace(., is.na(.), "SS_22"))

get_gene_type <- function(x){key_mIrna_pIrna %>%
  filter(gene_id== x) %>%
  .$seq_RNA}

test_mut_reads %>% head %>%
  mutate_at(vars(starts_with("V")), ~map_chr(.,get_gene_type))
```