

wind: wORKFLOW FOR PiRNAs AnD BEYONd

Computational workflow for Data Exploration resulted from smallRNA-seq

Constantinos Yeles (Konstantinos Geles)

Fri Nov 13 2020

Contents

1. Load libraries	1
2. Directory generation for the resulted files	2
i. Add date	2
ii. Make the directory for the results of the exploratory data analysis	2
3. Import the salmon files	2
4. Make or import the targets file.	2
5. Make a DGElist object for salmon	3
6. Import the FeatureCounts object and make a DGElist object	3
7. Create biodetection plot with NOISeq	4
8. Create the design matrix	5
9. Perform various Filtering Methods: EdgeR, NOISeq	5
10. Histogram before and after filtering of data	6
11. Normalization	6
12. Make h-clustering	7
13. Make MDS plot	8
14. Make PCA plot	9
15. Compare groups between FeatureCounts and salmon results	10
16. Make histogram of length	11
17. Sequence logos	13
18. Reads info and histograms	13
i. extract the information of length for reads	13
ii. make a txt file with the info	14
iii. ggplot for histograms	14

1. Load libraries

```
suppressPackageStartupMessages({
  library('tidyverse')
  library('data.table')
  library('plyranges')
  library('tximport')
  library('edgeR')
  library('NOISeq')
  library('rafalib')
  library('pheatmap')
  library('RColorBrewer')
})
```

2. Directory generation for the resulted files

i. Add date

Used as an identifier for the folder

```
todate <- format(Sys.time(), "%d_%b_%Y")
```

ii. Make the directory for the results of the exploratory data analysis

```
my_basename <- "GSE68246" ## INPUT name of the folder
my_exp <- "br_CSC_cancer" ## INPUT name of the analysis
genome_input <- "hg38" ## INPUT genome version here
my_exp_sal <- "salmon"
my_exp_fc <- "featureCounts"
dat_path <- str_glue("{my_basename}/ExpDatAnalysis_{my_exp}_{genome_input}_{todate}/{c(my_exp_sal,my_exp_fc)}")
dat_path %>% map(~dir.create(str_glue("./{.x}"), recursive = TRUE))
```

3. Import the salmon files

```
# load salmon files----
files_salm <- list.files(path = my_basename, pattern = ".sf",
  recursive = TRUE, full.names = TRUE)

names(files_salm) <- files_salm %>%
  str_remove("/quant.sf") %>% basename %>% str_remove(".trimmed.+")

# tximport-----
txi <- tximport::tximport(files_salm, type = "salmon",
  txOut = TRUE, countsFromAbundance = "lengthScaledTPM")
```

4. Make or import the targets file.

If you use a public dataset you can download it from SRA RUN selector Metadata or you can create it. It has to have at least three columns with the column names: “sample_name”, “group”, “batch” Here we show both ways although we need to manually add some columns in the metadata of SRA run selector.

```

# make the targets object ----
# INPUT targets file
targets_file <- tibble(  ## INPUT targets file here
  file_name = files_salm,
  sample_name = names(files_salm),
  group = as_factor(c(rep("MCF_7_Monolayer",3),rep("MCF_7_Spheroid",3))),
  batch = as_factor(c(1:3,1:3))
)

```

5. Make a DGElist object for salmon

```

# DGElist
# from https://bioconductor.org/packages/release/bioc/vignettes/tximport/inst/doc/tximport.html
# we follow the instructions to import for edgeR
cts <- txi$counts
normMat <- txi$length

# Obtaining per-observation scaling factors for length, adjusted to avoid
# changing the magnitude of the counts
normMat <- normMat/exp(rowMeans(log(normMat)))
normCts <- cts/normMat

# Computing effective library sizes from scaled counts, to account for
# composition biases between samples
eff.lib <- calcNormFactors(normCts) * colSums(normCts)

# Combining effective library sizes with the length factors, and calculating
# offsets for a log-link GLM
normMat <- sweep(normMat, 2, eff.lib, "*")
normMat <- log(normMat)

# Creating a DGEList object for use in edgeR.
dgl_salmon <- DGEList(cts, samples = targets_file) %>%
  scaleOffset( normMat) %>%
  write_rds(str_glue("{dat_path[1]}/dgl_edgeR_salmon.rds"))
# remove objects.
rm(cts, normCts, normMat, txi)

```

6. Import the FeatureCounts object and make a DGElist object

```

# load the rds from featureCounts----
#INPUT rds featureCounts
fc <- list.files(path = my_basename, recursive = TRUE,
  pattern = ".+counts+.rds",
  full.names = TRUE) %>%
  read_rds()

# write the matrix for the analysis, annotation stats-----
colnames(fc$counts) <- targets_file$sample_name

```

```

fc$counts %>% as_tibble(rownames = "sRNA") %>%
  write_tsv(str_glue("{dat_path[2]}/raw_reads_fc.txt"))
fc$annotation %>%
  as_tibble() %>%
  write_tsv(str_glue("{dat_path[2]}/annotation_fc.txt"))
fc$stat %>%
  as_tibble() %>%
  write_tsv(str_glue("{dat_path[2]}/stats_fc.txt"))
dgl_fc <- DGEList(counts = fc$counts,
  samples = targets_file,
  lib.size = colSums(fc$counts),
  norm.factors = rep(1,ncol(fc$counts)))

# give colours to samples ----
pal1 <- tibble(value = c("#E50B36", "#D2CA1D", "#79402E", "#FF0000",
  "#D8B70A", "#02401B", "#046C9A", "#5B1A18")) %>%
  dplyr::slice(1:length(levels(as_factor(targets_file$group)))) %>%
  mutate(
    group = as_factor(levels(as_factor(targets_file$group))))
dgl_fc$colours <- as_factor(inner_join(dgl_fc$samples, pal1, by= "group")$value)

# remove objects ----
rm(pal1)

```

7. Create biodection plot with NOISeq

```

mybiotypes <- fc$annotation %>%
  mutate(gene_type = gene_type %>% str_remove(";.+")) %>%
  select(GeneID, gene_type) %>%
  column_to_rownames("GeneID")

function_Noiseq_plots <- function(exp_data, plot_path){
  mydata <- NOISeq::readData(data = exp_data,
    factors = as.data.frame(targets_file),
    biotype = mybiotypes)
  mybiodection <- dat(mydata, k = 0, type = "biodection")
  pdf(str_glue("{plot_path}/NOISeq_biodection_exprs_{todate}_{str_remove(plot_path, './')}.pdf"))
  seq(ncol(exp_data)) %>% map(~explo.plot(mybiodection, samples = .x), plottype = "boxplot")
  dev.off()
  mycountsbio <- dat(mydata, factor = NULL, type = "countsbio")
  pdf(str_glue("{plot_path}/NOISeq_countsbio_{todate}_{str_remove(plot_path, './')}.pdf"))
  seq(ncol(exp_data)) %>% map(~explo.plot(mycountsbio,
    samples = .x, plottype = "boxplot"))
  dev.off()
}

list( "salmon" = dgl_salmon$counts, "fc" = fc$counts) %>%
  map2(.y = dat_path, ~function_Noiseq_plots(.x, .y))

```

8. Create the design matrix

```
design <- model.matrix(~0 + targets_file$group)
colnames(design) <- colnames(design) %>%
  str_remove("targets_file\\$group")
rownames(design) <- targets_file$sample_name
design_2 <- model.matrix(~0 + targets_file$group + targets_file$batch)
colnames(design_2) <- colnames(design_2) %>%
  str_remove("targets_file\\$group") %>%
  str_remove("targets_file\\$batch")
rownames(design_2) <- targets_file$sample_name
```

9. Perform various Filtering Methods: EdgeR, NOIseq

```
function_filtering <- function(dgl_data, data_path){
  # filtering with NOISeq -----
  noifil <- list("cpm" = 1L, "Prop" = 3L) %>%
    map(~NOISeq::filtered.data(dgl_data$counts,
      factor = targets_file$group,
      norm = FALSE,
      method = .x, cv.cutoff = 100, cpm = 1)
    )

  noifil %>%
    names %>%
    map( ~ dgl_data[rownames(dgl_data$counts) %in%
      rownames(noifil[.x]),,keep.lib.sizes = FALSE] %>%
      write_rds(str_glue("{data_path}/dgl_{.x}_filt_{str_remove(data_path, ' .+/' )}.rds"))
    )

  # filter with EdgeR ----
  keep.exprs <- filterByExpr.DGEList(dgl_data, design = design)
  keep.exprs_2 <- filterByExpr.DGEList(dgl_data, design = design_2)
  dgl_filt <- dgl_data[keep.exprs,,keep.lib.sizes=FALSE] %>%
    write_rds(str_glue("{data_path}/dgl_filt_nobatch_{str_remove(data_path, ' .+/' )}.rds"))
  dgl_filt_2 <- dgl_data[keep.exprs_2,,keep.lib.sizes=FALSE] %>%
    write_rds(str_glue("{data_path}/dgl_filt_batch_{str_remove(data_path, ' .+/' )}.rds"))

  features_NOIS <- map(noifil, ~ .x %>%
    rownames() %>%
    enframe(name = NULL))
  features_edgeR <- map(list(dgl_filt, dgl_filt_2) , ~ .x %>%
    rownames() %>%
    enframe(name = NULL)) %>%
    set_names("no_batch", "batch")

  common_edgeR_nobatch <- map2(features_edgeR[1], features_NOIS, ~ .x %>%
    inner_join(.y))
  common_edgeR_batch <- map2(features_edgeR[2], features_NOIS, ~ .x %>%
    inner_join(.y))

  filter_info <- tibble(
```

```

"features" = c("Starting_features:", "edgeR_nobatch_filter:",
  "edgeR_batch_filter:",
  "NOISeq_1cpm_filter:",
  "common_with_edgeR_nobatch:", "common_with_edgeR_batch:",
  "NOISeq_Proportion_filter:",
  "common_with_edgeR_nobatch:", "common_with_edgeR_batch:"
),
"number_of_features" = c(nrow(dgl_data$counts), nrow(dgl_filt$counts),
  nrow(dgl_filt_2$counts),
  nrow(noifil[[1]]),
  nrow(common_edgeR_nobatch[[1]]), nrow(common_edgeR_batch[[1]]),
  nrow(noifil[[2]]),
  nrow(common_edgeR_nobatch[[2]]), nrow(common_edgeR_batch[[2]]))
)
) %>%
  write_tsv(str_glue("{data_path}/filtering_info_{str_remove(data_path,'./')}").txt"))
dgl_filt
}

filtered_dgls <- list("salmon" = dgl_salmon, "fc" = dgl_fc) %>%
  map2(.y = dat_path, ~function_filtering(.x,.y))

```

10. Histogram before and after filtering of data

```

function_hist <- function(dgl_data, dgl_fil_data, plot_path){
  AveLogCpm_Raw_Data <- aveLogCPM(dgl_data)
  AveLogCpm_Filtered_Data <- aveLogCPM(dgl_fil_data)
  pdf(str_glue("{plot_path}/histogram_plot_{todate}_{str_remove(plot_path,'./')}").pdf"))
  hist(AveLogCpm_Raw_Data)
  hist(AveLogCpm_Filtered_Data)
dev.off()
}
list(list("salmon" = dgl_salmon, "fc" = dgl_fc),
  filtered_dgls, dat_path) %>%
  pmap(function_hist)

```

11. Normalization

```

function_EDA_RLE <- function(data,name){EDASeq::plotRLE(data,
  col = as.character(dgl_fc$colours),
  outline=FALSE, las=3,
  ylab="Relative Log Expression",
  cex.axis=1, cex.lab=1, main = str_glue("{name}"))
  legend("topright",
  legend= levels(as_factor(dgl_fc$samples$group)),
  fill = levels(as_factor(dgl_fc$colours)),
  bty="n",
  cex = 0.5, inset = c(.01,.01))
}

```

```

function_norm <- function(dgl_fil_data, data_path){
  # edgeR ----
  norm_method <- list("none", "TMM", "TMMwsp", "RLE") %>%
    set_names(.)
  edger_norm <- map(norm_method, ~calcNormFactors(dgl_fil_data, method = .x))
  # voom ----
  pdf(str_glue("{data_path}/voom_plots_{str_remove(data_path,'./')}.pdf"))
  voom_norm <- edger_norm[1:3] %>%
    map2(.y = c("quantile", rep("none",2)),
      ~voom(.x, design = design,
        plot = TRUE, normalize.method = .y)) %>%
    set_names("voom_Quantile", "voom_TMM", "voom_TMMwsp")
  dev.off()
  # voom with quality weights ----
  pdf(str_glue("{data_path}/voom_quality_weights_plots_{str_remove(data_path,'./')}.pdf"))
  voom_norm_QW <- edger_norm[1:3] %>%
    map2(.y = c("quantile", rep("none",2)),
      ~voomWithQualityWeights(.x, design = design,
        plot = TRUE, normalize.method = .y)) %>%
    set_names("voomQW_Quantile", "voomQW_TMM", "voomQW_TMMwsp")
  dev.off()
  # list of normalized data ----
  norm_list <- c(edger_norm %>% map(~cpm(.x, normalized.lib.sizes = TRUE)),
    list(
      "voom_Quantile" = 2^voom_norm[[1]]$E,
      "voom_TMM" = 2^voom_norm[[2]]$E,
      "voom_TMMwsp" = 2^voom_norm[[3]]$E,
      "voomQW_Quantile" = 2^voom_norm_QW[[1]]$E,
      "voomQW_TMM" = 2^voom_norm_QW[[2]]$E,
      "voomQW_TMMwsp" = 2^voom_norm_QW[[3]]$E))
  pdf(str_glue("{data_path}/RLE_plots_{str_remove(data_path,'./')}.pdf"))
  norm_list %>%
    imap(~function_EDA_RLE(.x,.y))
  dev.off()
  norm_list[2:4] %>% imap(~.x %>%
    as_tibble(rownames = "GeneIDs") %>% write_tsv(path =
      str_glue("{data_path}/norm_cpm_{.y}_{str_remove(data_path,'./')}.txt")))
  c(edger_norm, voom_norm, voom_norm_QW)
}

norm_dgls <- filtered_dgls %>%
  map2(.y = dat_path, ~function_norm(.x, .y))
# save the list with all normalized values (edgeR and limma)-----
norm_dgls %>%
  map2( .y = dat_path,
    ~write_rds(.x,file = str_glue("{.y}/list_norm_dgls_{str_remove(.y,'./')}.rds")))

```

12. Make h-clustering

```

function_clust <- function(dgl_norm_data, plot_path){
  hc_methods <- c("ward.D2",

```

```

      "complete",
      "average")

list_distc <- c(dgl_norm_data[1:4] %>%
  map(~cpm(.x, normalized.lib.sizes = TRUE, log=TRUE, prior.count=5)),
  list("voom_Quantile" = dgl_norm_data[[5]]$E,
    "voom_TMM" = dgl_norm_data[[6]]$E,
    "voom_TMMwsp" = dgl_norm_data[[7]]$E,
    "voomQW_Quantile" = dgl_norm_data[[8]]$E,
    "voomQW_TMM" = dgl_norm_data[[9]]$E,
    "voomQW_TMMwsp" = dgl_norm_data[[10]]$E)) %>% map(~dist(t(.x)))
#pheatmap start
list_distc_mat <- list_distc %>% map(~as.matrix(.x))
colours_pheat <- colorRampPalette(rev(brewer.pal(9, "Blues")))(255)
pdf(str_glue("{plot_path}/distance_matrix_hclust_{str_remove(plot_path, '.*+')}.pdf"))
list_distc_mat %>% imap(~pheatmap(.x,
  clustering_distance_rows = "euclidean",
  clustering_distance_cols = "euclidean",
  col = colours_pheat,
  main = str_glue({.y})))
dev.off()
#pheatmap end
#list_distc <- log_cpm %>% map(~dist(t(.x)))
list_hc <- sapply(hc_methods, function(x) map(list_distc, ~hclust(.x, method = x)))
names(list_hc) <- rep(rownames(list_hc), times = ncol(list_hc))

pdf(str_glue("{plot_path}/hierarchic_clust_{str_remove(plot_path, '.*+')}.pdf"))
for (i in seq_along(list_hc)) {
  rafalib::myplclust(list_hc[[i]],
    lab.col = as.character(dgl_fc$colours),
    xlab = NULL,
    main = str_glue("{enframe(list_hc[[i]])$value[7]} - {enframe(list_hc[[i]])$value[5]} - {names(list_hc)[i]}"),
    legend("topright",
    legend = levels(dgl_fc$samples$group),
    fill = levels(dgl_fc$colours),
    bty="n",
    cex = 0.9)
  }
dev.off()
}

map2(norm_dgls, dat_path, ~function_clust(.x,.y))

```

13. Make MDS plot

```

function_MDS <- function(dgl_norm_data, plot_path){
  par(mar=c(6,5,2,1)+ 0.1)
  pdf(str_glue("{plot_path}/MDS_plot_{str_remove(plot_path, '.*+')}.pdf"))
  plotMDS(dgl_norm_data$TMM,
    labels = dgl_fc$samples$sample_name,
    pch = 10,

```



```

        cex = 0.7,
        col = as.character(dgl_fc$colours), dim.plot = c(1,2))
legend("topright",
      legend = levels(dgl_fc$samples$group),
      fill = levels(dgl_fc$colours),
      bty="n",
      cex = 1.5, inset = c(.01,.09))
map2(c(3,1,2,2),c(4,3,3,4),
~plotMDS(dgl_norm_data$TMM, labels = dgl_fc$samples$sample_name, pch = 10,
  cex = 0.7,
  col = as.character(dgl_fc$colours),
  dim.plot = c(.x,.y),
  main = str_glue("MDS plot {names(dgl_norm_data[2])}"))
)
dev.off()
}
map2(norm_dgls, dat_path, ~function_MDS(.x,.y))

```

14. Make PCA plot

```

## modified from DESeq2::plotPCA
## https://github.com/mikelove/DESeq2/blob/master/R/plots.R

function_PCA <- function(dgl_norm_data, plot_path, norm_method = "TMM", ntop = 500){
  library(DESeq2)
  # calculate the variance for each gene
  rv <- rowVars(dgl_norm_data[[norm_method]]$counts)

  # select the ntop genes by variance
  select <- order(rv, decreasing=TRUE)[seq_len(min(ntop, length(rv)))]

  # perform a PCA on the data in dgl_norm_data[[norm_method]]$counts for the selected genes
  pca <- prcomp(t(dgl_norm_data[[norm_method]]$counts[select,]))

  # the contribution to the total variance for each component
  percentVar <- pca$sdev^2 / sum( pca$sdev^2 )

  # create a new grouping factor
  group <- dgl_norm_data[[norm_method]]$samples$group

  # assembly the data for the plot
  d <- data.frame(PC1=pca$x[,1], PC2=pca$x[,2], group=group,
    name=colnames(dgl_norm_data[[norm_method]]$counts))

  # create batch if exist
  if(dgl_norm_data[[norm_method]]$samples$batch){
    d$batch <- dgl_norm_data[[norm_method]]$samples$batch %>% as_factor
    p <- ggplot(data=d,
      aes_string(x="PC1",
        y="PC2",
        color="batch",

```

```

                                shape="group"))
}else{
p <- ggplot(data=d, aes_string(x="PC1", y="PC2", shape="group"))
}

p <- p +
  geom_point(size=3) +
  xlab(paste0("PC1: ",round(percentVar[1] * 100),"% variance")) +
  ylab(paste0("PC2: ",round(percentVar[2] * 100),"% variance")) +
  coord_fixed()+
  theme_minimal()+
  labs(title=str_glue("PCA plot {norm_method}"))+
  theme(plot.title = element_text(hjust = 0.5),
        axis.text.x = element_text( face = "bold"),
        axis.text.y = element_text( face = "bold"),
        aspect.ratio = 1)

# pdf
par(mar=c(6,5,2,1)+ 0.1)
pdf(str_glue("{plot_path}/PCA_plot_{norm_method}_{str_remove(plot_path,'./')}").pdf")
print(p)
dev.off()
}

map2(norm_dgls, dat_path, ~function_PCA(.x,.y))

```

15. Compare groups between FeatureCounts and salmon results

```

function_comp_groups <- function(dgl_norm_data, tool){
  grouped_cpm <- dgl_norm_data$TMM %>%
    cpmByGroup.DGEList
  # keep <- rowSums(grouped_cpm > 1) > 0
  # grouped_cpm[keep,] %>%
    grouped_cpm %>%
    as_tibble(rownames = "sncRNA") %>%
    rename_at(vars(-matches("sncRNA")), list(~str_c(.,tool)))
}

comp_FC_sal <- map2(norm_dgls, list("_salmon", "_fc"), ~function_comp_groups(.x,.y))

#comp_FC_sal <- map2(list("salmon" = norm_dgls_sal, "FC" = norm_dgls_FC),
#  # list("_salmon", "_FC"), ~function_comp_groups(.x,.y))

#comp_FC_sal$salmon %>%
#  inner_join(comp_FC_sal$fc, by = "sncRNA") %>%
#  write_tsv(str_glue("{str_remove(dat_path[1], './')}salmon_FC_1cpm_common_grouped.txt"))

salmon_FC_cpm_union_grouped <- comp_FC_sal$salmon %>%
  left_join(mybiotypes %>% as_tibble(rownames="sncRNA")) %>%
  full_join(comp_FC_sal$fc, by = "sncRNA") %>%
  select(sncRNA, gene_type, everything()) %>%

```

```

write_tsv(str_glue("{str_remove(dat_path[1], '/salmon|/featureCounts')}salmon_FC_cpm_union_grouped.txt")

# piCK the 100 top expressed molecules between FC salmon and all groups -----
all_exprs_cpm_TMM <- list.files(recursive = T,
  pattern = "norm_cpm_TMM_",
  full.names = T) %>%
vroom::vroom(id = "method") %>%
mutate(method = method %>% basename() %>% str_remove("norm_cpm_TMM_") %>% str_remove(".txt"))

salmon_FC_cpm_union_grouped_top <- salmon_FC_cpm_union_grouped %>%
  filter(gene_type == "piRNA") %>%
  arrange(across(.fns = dplyr::desc,
    .cols = ends_with(c("salmon", "fc")))) %>%
  slice_head(n = 100)

all_exprs_cpm_TMM %>%
mutate(method = if_else(method == "featureCounts",
  true = "fc",
  false = "salmon")) %>%
pivot_longer(cols = !c(method, GeneIDs)) %>%
  unite(col = "sample", c(name, method)) %>%
pivot_wider(names_from = "sample",
  values_from = "value") %>%
column_to_rownames("GeneIDs") %>%
.[salmon_FC_cpm_union_grouped_top$sncRNA,] %>%
as_tibble(rownames = "smallRNA") %>% write_tsv(str_glue("{str_remove(dat_path[1], '/salmon|/featureCounts')}smallRNA_fc_salmon.txt"))

```

16. Make histogram of length

```

annot_tbl <- read_gff2("../genome_transc_human/ncRNA_transcripts_100bp_RNA_Central_piRNAbank_hg38.gtf")
as_tibble %>%
select(-c(type, score, phase)) %>%
dplyr::rename("length_w" = width)

function_prep_hist <- function(dgl_norm_data, annot_gtf){
  prep_hist <- annot_gtf %>%
    distinct(gene_id, .keep_all = T) %>%
    filter(gene_id %in% rownames(dgl_norm_data$TMM))
}

smallRNA_seqs <- map2(norm_dgls, list(annot_tbl, annot_tbl), ~function_prep_hist(.x, .y))

smallRNA_seqs[["fc"]] %>% # create one also per group
dplyr::count(gene_type, sort = T) %>%
dplyr::rename("fc_n" = n) %>%
full_join( smallRNA_seqs[["salmon"]] %>%
dplyr::count(gene_type, sort = T) %>%
dplyr::rename("salmon_n" = n)) %>%
full_join(
  smallRNA_seqs[["fc"]] %>%
  inner_join(smallRNA_seqs[["salmon"]] ) %>%

```

```

dplyr::count(gene_type, sort = T) %>%
dplyr::rename("common_n" = n) %>%
full_join(
smallRNA_seqs[["fc"]] %>%
  anti_join(smallRNA_seqs[["salmon"]]) %>%
dplyr::count(gene_type, sort = T) %>%
dplyr::rename("unique_FC_n" = n) %>%
full_join(
smallRNA_seqs[["salmon"]] %>%
  anti_join(smallRNA_seqs[["fc"]]) %>%
dplyr::count(gene_type, sort = T) %>%
dplyr::rename("unique_salmon_n" = n) %>%
write_tsv(str_glue("{str_remove(dat_path[1], '/salmon|/featureCounts')} /stats_gene_types_ids.txt"))

# per group summary ----
groups <- targets_file$group %>% levels() %>% set_names()

salmon_groups <- groups %>% map(~comp_FC_sal$salmon %>%
  filter_at(vars(matches(!str_c(.x, "_salmon"))),
    any_vars(. > 0)) %>%
  select(sncRNA) %>%
  mutate(!str_c("group_", .x) := .x)
) %>% purrr::reduce(full_join) %>%
mutate(salmon = "salmon") %>%
left_join(smallRNA_seqs$salmon, by = c("sncRNA" = "gene_id")) %>%
select(-c(seqnames:end, strand, source, sRNA_id, sRNA_id2)) %>%
pivot_longer(cols = starts_with("group_"), values_to = "rank_salmon"
) %>% select(-name)

FC_groups <- groups %>% map(~comp_FC_sal$fc %>%
  filter_at(vars(matches(!str_c(.x, "_fc"))),
    any_vars(. > 0)) %>%
  select(sncRNA) %>%
  mutate(!str_c("group_", .x) := .x)
) %>%
purrr::reduce(full_join) %>%
mutate(FeatureCount = "FeatureCount") %>%
left_join(smallRNA_seqs$fc, by = c("sncRNA" = "gene_id")) %>%
select(-c(seqnames:end, strand, source, sRNA_id, sRNA_id2)) %>%
pivot_longer(cols = starts_with("group_"), values_to = "rank_FC"
) %>% select(-name)

## histograms ----
pdf(str_glue("{str_remove(dat_path[1], '/salmon|/featureCounts')} /length_histogram.pdf"))
map(FC_groups$gene_type %>% as_factor() %>% levels(),
  ~salmon_groups %>%
  full_join(FC_groups,
    by = c("sncRNA", "seq_RNA",
      "gene_type", "length_w")) %>%
  filter(gene_type == .x) %>%
  pivot_longer(c(salmon, FeatureCount),
    "tool", values_to = "Quantification") %>%
  pivot_longer(c(rank_salmon, rank_FC), "rank", values_to = "groups") %>%

```

```

filter(!is.na(Quantification),!is.na(groups)) %>%
  ggplot() +
  geom_bar(mapping = aes(x = factor(length_w), fill = Quantification),
    position = "dodge") +
  facet_wrap(~ groups, ncol = 2) +
  scale_x_discrete(name = 'length') +
  scale_y_continuous(labels = scales::comma) +
  ggtitle(.x) +
  coord_flip()
)
dev.off()

```

17. Sequence logos

```

# sequences logos -----
library(ggseqlogo)

pdf(str_glue("{str_remove(dat_path[1], '/salmon|/featureCounts')}/piRNA_logos_FC_salmon.pdf"))
#salmon
groups %>% map2(list(salmon_groups),
  ~.y %>%
    filter(gene_type == "piRNA") %>%
    filter_at(vars(starts_with("rank")), any_vars(. == .x)) %>%
    .$seq_RNA %>%
    str_sub(1,15) %>%
    ggseqlogo(method = 'prob', font="roboto_regular") +
    ggtitle(str_glue("Salmon_{.x}")))
#featurecounts
groups %>% map2(list(FC_groups),
  ~.y %>%
    filter(gene_type == "piRNA") %>%
    filter_at(vars(starts_with("rank")), any_vars(. == .x)) %>%
    .$seq_RNA %>%
    str_sub(1,15) %>%
    ggseqlogo(method = 'prob', font="roboto_regular") +
    ggtitle(str_glue("FeatureCounts_{.x}")))
dev.off()

```

18. Reads info and histograms

i. extract the information of length for reads

```

## filter for reads of 15-49 bases
for file in my_data/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 | grep -F -w -f 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) and saving in: ${where_to_save}/${samp}_hist.txt";
samtools view -@ 6 ${file} | awk 'BEGIN{FS=OFS="\t"}{print length($10),$18}' | sed 's/XT:Z://g' | sort
echo "end:$(date)";
done

## find length of reads from salmon
for file in my_data/quants/*.fastq.gz.bam;
do
where_to_save=`dirname ${file}`;
regex=`basename ${file}`;
samp="${regex%*.trimmed*}";
echo "Processing sample ${samp} start: $(date) and saving in: ${where_to_save}/${samp}_hist_allRNA.txt";
samtools view -@ 2 ${file} | awk 'BEGIN{FS=OFS="\t"}{print $1,length($10),$3}' | sort -k1,1 | bedtools g
echo "end:$(date)";
done

## if was to make only for piRNA
samtools view -@ 6 ${file} | awk 'BEGIN{FS=OFS="\t"}{print length($10),$3}' | grep -F -w -f my_data/piRNA

```

ii. make a txt file with the info

```

library(tidyverse)
hist_all_RNA <- list.files(path = "../genome_transc_human/quants",
                           pattern = "hist_allRNA.txt",
                           full.names = T) %>%
  vroom::vroom(id = "file", col_names = c("read_count", "read_length", "smallRNA")) %>%
  mutate(file = basename(file) %>% str_remove("_hist_all.+"))

```

iii. ggplot for histograms

```

library(tidyverse)
library(plyranges)
#piRNAs_hist <- read_tsv("piRNAs_hist.txt")
# gtf_piB_RCentr <- read_gff2("mouse_data/ncRNA_transcripts_100bp_RNA_Central_piRNAbank_mm10.gtf") %>%
  as_tibble() %>%
  select(gene_id, gene_type) %>%
  distinct(gene_id, .keep_all = T) %>%

```

```

mutate(gene_type = as_factor(gene_type))
# FC and salmon files -----
hist_files_fc <- list.files(path = "mouse_datasets/star_results",
                           pattern = "_hist.txt",
                           recursive = TRUE, full.names = TRUE) %>%

  set_names(. %>%
            basename() %>%
            str_remove("_hist.txt")
            )

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

  set_names(. %>%
            basename() %>%
            str_remove("_hist_allRNA.txt")
            )

#FC reads info----
## add the regex of smallRNA names
smallRNA_Categ_regex <- levels(gtf_piB_RCentr$gene_type) %>%
  set_names(.) %>%
  map(~gtf_piB_RCentr %>%
    filter(gene_type == .x) %>%
    .$gene_id %>%
    str_c(collapse = "|")) %>%
  c(., "Not_assigned" = "Not_assigned")

## function to filter a df of mapped reads per small RNA category using regex
reads_info_fc <- function(df, Categ_regex_list){
  Categ_regex_list %>% #regex with rna_names "mmu_piR_039595|mmu_piR_039148"
  imap(~df %>% # for each category summarise the reads
    filter(str_detect(sncRNA,.x)) %>%
    summarise(!str_c(.y, "_Reads") := sum(Reads)) %>%
    add_column(Category = "Reads")
  ) %>%
  purrr::reduce(left_join) %>%
  left_join(df %>%
    filter(Alignment!="Not_assigned") %>%
    group_by(Alignment) %>%
    summarise(S=sum(Reads)) %>%
    mutate( Category = "Reads") %>%
    pivot_wider(names_from = Alignment, values_from = S)) %>%
  mutate(Reads_in_analysis = Not_assigned_Reads + Multimapped + Unique,
    Mapped_Reads = Multimapped + Unique,
    Mapped_Reads_perc =
    as.character(round((Mapped_Reads/Reads_in_analysis)*100, digits = 2)) %>%
    str_c(., "%")) %>%
  select(Category, Reads_in_analysis, Mapped_Reads,
    Mapped_Reads_perc,
    Unique,
    Multimapped,
    Not_assigned_Reads, everything())

```

```

}

## read the hist files and mutate them
df_list <- hist_files_fc %>%
  map(~read_tsv(., col_names = c("Reads", "Length", "sncRNA")) %>%
    mutate(Length = as_factor(Length),
           sncRNA = if_else(is.na(sncRNA), "Not_assigned", sncRNA),
           Alignment = case_when(
             sncRNA == "Not_assigned" ~ "Not_assigned",
             str_detect(sncRNA, ",") ~ "Multimapped" ,
             TRUE ~ "Unique" )
          )
    )

## create the data
FC_reads_info <- df_list %>%
  imap(~reads_info_fc(., smallRNA_Categ_regex) %>%
    mutate(Category = .y)) %>%
  purrr::reduce(bind_rows)

dgl_norm$featureCounts$TMM$counts %>%
  as_tibble(rownames = "smallRNA") %>%
  filter(smallRNA %in% annot_tbl$gene_id) %>% # annot_tbl <- gtf_piB_RCentr %>% filter(gene_type == "piR
  pivot_longer(~smallRNA) %>%
  mutate(piRNA = if_else(value > 0, 1, 0)) %>%
  group_by(name) %>%
  summarise(piRNAs_in_sample = sum(piRNA)) %>%
  dplyr::rename(Category = name) %>%
  left_join(FC_reads_info) %>% # save them on a txt
write_tsv("mouse_datasets/FC_reads_info.txt")

# FeatureCounts histograms of piRNAs-----
piRNA_DF_hist <- df_list %>%
  bind_rows(.id = "Sample") %>%
  filter(sncRNA != "Not_assigned") %>%
  filter(str_detect(sncRNA, smallRNA_Categ_regex[["piRNA"]]))

#FC reads hist piRNA----
piRNA_DF_hist %>%
  group_by(Sample, Length, Alignment) %>%
  summarise(Reads = sum(Reads)) %>%
  ggplot() +
  geom_bar(mapping =
    aes(x = Length,
        y = Reads,
        fill = Alignment),
    stat = "identity") +
  scale_y_continuous(labels = scales::comma) +
  coord_cartesian(ylim = c(0, 2000000)) +
  theme_bw() +
  facet_wrap(~Sample, ncol = 3) +

```



```

    theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust=1))

pdf(str_glue("mouse_datasets/histograms_piRNA_reads_facets_fc.pdf"))
df_list %>% imap(~.x %>%
  filter(str_detect(sncRNA, smallRNA_Categ_regex[["piRNA"]])) %>%
  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_bw() +
    facet_grid(Alignment ~ .) +
    ggtitle(str_glue(" Histogram of {.y} sample"))+
    theme(plot.title = element_text(hjust = 0.5),
      axis.text.x = element_text(angle = 45, vjust = 1, hjust=1))
)
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))

```